

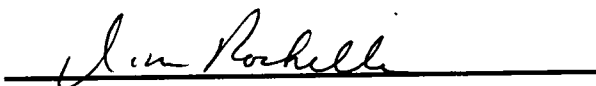
To the Graduate Council:

I am submitting herewith a thesis written by Steven Dwain Zimmermann entitled "Design and Implementation of an Electronic Pan/Tilt/Rotate/Magnify Camera System for Wide-Angle Viewing Applications." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.



Donald W. Bouldin, Major Professor

We have read this thesis  
and recommend its acceptance:



Accepted for the Council:



Associate Vice Chancellor  
and Dean of the Graduate School

# STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at the University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library, Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made.

Requests for permission for extensive quotation from or reproduction of this thesis in whole or in parts may be granted by the copyright holder. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature

Steve Zimmermann

Date

12/1/92

**DESIGN AND IMPLEMENTATION OF AN  
ELECTRONIC PAN/TILT/ROTATE/MAGNIFY  
CAMERA SYSTEM FOR WIDE-ANGLE VIEWING  
APPLICATIONS**

**A Thesis Presented for the  
Master of Science  
Degree**

**The University of Tennessee, Knoxville**

**Steven D. Zimmermann  
December 1992**

Copyright © Steven Dwain Zimmermann, 1992  
All rights reserved

# **DEDICATION**

This thesis is dedicated to my family

Sherry Denise, Stephanie Deanne, and Sarah Danielle Zimmermann

and to my parents

Mr. Johnny L. Zimmermann and Ms. Linda F. Zimmermann

## ACKNOWLEDGMENTS

I would like to thank and show appreciation to many individuals who provided guidance and support during this endeavor. I thank the University of Tennessee Department of Electrical Engineering for being an outstanding center for higher education. I thank my committee members, Dr. Don Bouldin, Dr. Jim Rochelle, and Dr. Robert Bodenheimer, for their support and confidence during the course of this effort. I would like to thank TeleRobotics International and NASA Langley Research Center for providing the environment and the opportunity, SBIR contract number NAS1-18855, respectively, that allowed me to develop the Omniview camera system. I would especially like to express appreciation to my family without whose support, confidence, and enthusiasm this would not have been nearly as rewarding - my wife Sherry and daughters Stephanie and Sarah, and my parents John Lee and Linda Zimmermann.

## ABSTRACT

A camera system for omni-directional image viewing applications that provides pan, tilt, zoom, and rotational orientation within a hemispherical field-of-view utilizing no moving parts has been developed. The imaging device is based on the effect that the image from a fisheye lens, which produces a circular image of an entire hemispherical field-of-view, can be mathematically corrected using high speed electronic circuitry. More specifically, an incoming fisheye image from any image acquisition source is captured in the memory of the device, a transformation is performed for the viewing region-of-interest and viewing direction, and a corrected image is output as a video image signal for viewing, recording, or analysis. As a result, this system can accomplish the functions of pan, tilt, rotation, and zoom throughout a hemispherical field-of-view without the need for any mechanical mechanisms. A programmable transformation processor provides flexible control over viewing situations. Multiple images, each with different image magnifications and pan-tilt-rotate parameters, can be obtained from a single camera. The image transformation processor can provide corrected images at frame rates compatible with RS-170 standard video equipment. The system can be used for many applications where a conventional mechanical pan-and-tilt orientation mechanism might be considered including inspection, monitoring, surveillance, and target acquisition. Omniview is ideal for multiple target acquisition and image stabilization in military applications due to its multiple image handling and fast response capabilities.

# TABLE OF CONTENTS

| CHAPTER                                             | PAGE      |
|-----------------------------------------------------|-----------|
| <b>1 INTRODUCTION .....</b>                         | <b>1</b>  |
| 1.1 Motivation .....                                | 1         |
| 1.2 Omniview Description .....                      | 2         |
| 1.3 Overview Of Chapter Contents .....              | 3         |
| 1.4 Summary .....                                   | 4         |
| <b>2 DEVELOPMENT OF THE MAPPING ALGORITHM .....</b> | <b>5</b>  |
| 2.1 Fisheye Lens Description .....                  | 5         |
| 2.2 Transformation Algorithm Development .....      | 8         |
| 2.3 Summary .....                                   | 14        |
| <b>3 OMNIVIEW IMAGE PROCESSOR DEVELOPMENT .....</b> | <b>15</b> |
| 3.1 Performance Considerations .....                | 15        |
| 3.2 General Purpose Approaches .....                | 17        |
| 3.3 XY Address LUT Implementation .....             | 19        |
| 3.4 Dedicated Hardware Approach .....               | 25        |
| 3.5 Summary .....                                   | 30        |
| <b>4 OMNIVIEW IMAGE PROCESSOR DESIGN .....</b>      | <b>32</b> |
| 4.1 Image Processor Architecture .....              | 32        |
| 4.2 Numerator Processor .....                       | 34        |
| 4.3 Denominator Processor .....                     | 41        |
| 4.4 High Speed Divider Implementation .....         | 47        |
| 4.5 Control Logic Design .....                      | 51        |
| 4.6 Summary .....                                   | 59        |
| <b>5 OMNIVIEW HOST PROCESSOR DESIGN .....</b>       | <b>61</b> |
| 5.1 Host Microprocessor Architecture .....          | 62        |
| 5.2 Overview Of Execution Environment .....         | 66        |
| 5.3 Overview of 80960 Bus Operation .....           | 69        |
| 5.4 Control Logic Implementation .....              | 74        |
| 5.5 Memory Implementation .....                     | 80        |

| CHAPTER                                          | PAGE       |
|--------------------------------------------------|------------|
| 5.6 Peripheral and I/O Interfaces .....          | 81         |
| 5.7 Summary .....                                | 85         |
| <b>6 OMNIVIEW VIDEO PROCESSOR DESIGN .....</b>   | <b>87</b>  |
| 6.1 VRAM Controller and Multi-Port Arbiter ..... | 89         |
| 6.2 Video Genlock Interface .....                | 93         |
| 6.3 Video Display Interface .....                | 97         |
| 6.4 Interpolation Filter .....                   | 98         |
| 6.5 Summary .....                                | 100        |
| <b>7 OPERATING SYSTEM SOFTWARE .....</b>         | <b>102</b> |
| 7.1 Monitor Program Description .....            | 102        |
| 7.2 Memory Support Considerations .....          | 103        |
| 7.3 Programming Tools .....                      | 106        |
| 7.4 Application Software .....                   | 106        |
| 7.5 Summary .....                                | 109        |
| <b>8 RESULTS .....</b>                           | <b>110</b> |
| 8.1 Software Omniview Demonstration .....        | 110        |
| 8.2 Wire-Wrap Development System .....           | 113        |
| 8.3 Omniview Prototype System .....              | 115        |
| 8.4 Omniview Display Control .....               | 120        |
| 8.5 Future Development .....                     | 121        |
| 8.6 Summary .....                                | 123        |
| <b>BIBLIOGRAPHY .....</b>                        | <b>125</b> |
| <b>VITA .....</b>                                | <b>128</b> |

# LIST OF FIGURES

| FIGURE                                                            | PAGE |
|-------------------------------------------------------------------|------|
| 1-1 Photograph of Prototype Omnview System .....                  | 3    |
| 2-1 Photograph of Typical Fisheye Image .....                     | 6    |
| 2-2 Diagram of Azimuth Angle Invariability Property .....         | 7    |
| 2-3 Diagram of Equidistant Projection Rule .....                  | 9    |
| 2-4 Coordinate Reference Frame Representation .....               | 10   |
| 3-1 XY Address LUT Block Diagram Approach .....                   | 20   |
| 3-2 VRAM Functional Block Diagram .....                           | 22   |
| 3-3 XY LUT Transform Graphical Representation .....               | 24   |
| 3-4 Omnview Functional Block Diagram .....                        | 26   |
| 3-5 Dedicate Numerator/Denominator Approach .....                 | 29   |
| 3-6 RSS Generator Functional Block Diagram .....                  | 30   |
| 4-1 Image Processor Functional Block Diagram .....                | 33   |
| 4-2 AIMS Functional Block Diagram .....                           | 35   |
| 4-3 UV Vector Generator Functional Block Diagram .....            | 42   |
| 4-4 Actel ACT™ 1 Logic Module Architecture .....                  | 44   |
| 4-5 Pythagoras Processor Functional Block Diagram .....           | 46   |
| 4-6 32-Bit Integer Divider Functional Block Diagram .....         | 48   |
| 4-7 Address Control Functional Block Diagram .....                | 50   |
| 4-8 Control Logic Block Diagram .....                             | 52   |
| 4-9 Image Processor Memory Map .....                              | 53   |
| 4-10 Field Programmable Controller Functional Block Diagram ..... | 54   |
| 4-11 Pipeline Control Timing Diagram .....                        | 57   |
| 4-12 CGC State Machine Diagram .....                              | 58   |
| 5-1 Host Processor Functional Block Diagram .....                 | 63   |
| 5-2 Intel 80960SB Microprocessor Internal Architecture .....      | 64   |
| 5-3 Intel 80960SB Execution Environment .....                     | 67   |
| 5-4 Intel 80960SB Logic Interface Diagram .....                   | 70   |
| 5-5 Intel 80690SB Bus State Diagram .....                         | 72   |

|      |                                                            |     |
|------|------------------------------------------------------------|-----|
| 5-6  | Burst Transaction Timing Diagram .....                     | 73  |
| 5-7  | Read/Write Timing Diagram (Non-Burst) .....                | 75  |
| 5-8  | Host Processor Memory Map .....                            | 77  |
| 5-9  | Bus Tracker Control PLD State Diagram .....                | 79  |
| 5-10 | KIO Functional Block Diagram .....                         | 83  |
| 6-1  | Video Processor Board Functional Block Diagram .....       | 88  |
| 6-2  | Video Processor Board Memory Map .....                     | 89  |
| 6-3  | Input Video Memory Buffer Interface Block Diagram .....    | 91  |
| 6-4  | Input VRAM Arbitration and SYNC Controller .....           | 92  |
| 6-5  | Video Genlock Interface Block Diagram .....                | 94  |
| 6-6  | Video Display Interface Block Diagram .....                | 98  |
| 6-7  | Interpolation Filter Block Diagram .....                   | 99  |
| 7-1  | Omniview System Software Memory MAP .....                  | 105 |
| 8-1  | Photograph of Software Demonstration .....                 | 112 |
| 8-2  | Photograph of Software Demonstration Parameter Input ..... | 112 |
| 8-3  | Photograph of Wire-Wrap Development Electronics .....      | 114 |
| 8-4  | Photograph of Demonstration Camera System .....            | 114 |
| 8-5  | Photograph of Prototype Omniview System .....              | 116 |
| 8-6  | Photograph of Image Processor Electronics Board .....      | 118 |
| 8-7  | Photograph of Host Processor Electronics Board .....       | 118 |
| 8-8  | Photograph of Video Processor Electronics Board .....      | 119 |
| 8-9  | Photograph of Omniview Electronics .....                   | 119 |
| 8-10 | Photograph of Single View Display .....                    | 121 |
| 8-11 | Photograph of Dual View Display .....                      | 122 |
| 8-12 | Photograph of Quad View Display .....                      | 122 |

# **CHAPTER 1**

## **INTRODUCTION**

Camera viewing systems are abundant in surveillance, inspection, security, and remote sensing applications. Remote viewing is critical for robotic manipulation tasks, and is often performed by cameras attached to mechanisms that provide the pan, tilt, zoom, and focus capabilities. Close viewing is necessary for detailed manipulation tasks while wide-angle viewing aids positioning of the robotic system to avoid collisions with the work space. The majority of these systems use either a fixed-mounted camera with a limited viewing field, or they utilize mechanical pan-and-tilt platforms and mechanized zoom lenses to orient the camera and magnify its image. These mechanisms can be large, unreliable, and may cause interference and collision with the environment. In applications where orientation of the camera and magnification of its image are required, the mechanical solution is large and can subtend a significant volume making the viewing system difficult to conceal or use in close quarters. Also, several cameras may be necessary to provide wide-angle viewing or complete coverage of the work space.

Camera viewing systems that use internal optics to provide wide viewing angles have been developed in order to minimize the size and volume of the camera and minimize the amount of intrusion into the viewing environment. These systems rely on the movement of either a mirror or prism to change the tilt-angle of orientation and provide mechanical rotation of the entire camera to change the pitch angle of orientation. Using this approach, the size of the camera orientation system can be minimized, but "blind spots" in the center of the view result. Also, these systems typically have no means of magnifying the image and or producing multiple images from a single camera.

### **1.1 MOTIVATION**

In order to minimize the size of the camera and orientation mechanism, the Omniview camera system was developed for remote viewing applications. The Omniview

camera utilizes fisheye lens optics and electronics processing to provide pan, tilt, zoom, and rotational capabilities within a hemispherical field-of-view with no moving parts. The Omniview camera approach is based on the properties of a fisheye lens which allows a complete hemispherical field-of-view to be captured, but with significant barrel distortion present in the image periphery. A high speed image transformation processor was developed that reconstitutes portions of the image with the correct perspective for display on an RS-170 standard format monitor. The Omniview camera system has several advantages over other camera systems. The Omniview implementation provides the capability to display multiple images simultaneously. A single omni-directional camera can provide several views from one camera location. The transformation is accomplished electronically, providing complete programmable control over viewing parameters. Image sizes, viewing directions, scale and offset, etc. may be adjusted to fit remote viewing needs. Since the fisheye image is symmetrical about the image center, the camera need not be oriented vertically and can be rotated electronically to accommodate different viewing situations.

## **1.2 OMNIVIEW DESCRIPTION**

A photograph of the prototype Omniview camera system is shown in Figure 1-1. The Omniview camera system consists of three main processing boards - the image processor, the host processor and video image buffer arbitration logic, and the video processor board. The electronics boards are 6 inch by 8 inch in size and are housed in a 6 inch by 8 inch by 2.5 inch electronic enclosure. The Omniview system includes five connectors that provide system interfaces. A computer control and remote control communication interface are provided by two front panel mounted DB-9 connectors. The communication interfaces are implemented with RS-422 type driver/receivers for long distance communication capability. A remote hand-held display controller provides the ability to control the Omniview system remotely through control port. A remote computer interface port is provided for use in automated or computerized operator interface applications and connects to either the left or right communication port. The Omniview system uses an external 120V ac powered +5V power supply that connects through a DB-9 type connector on the rear panel of the Omniview system. A genlock input interface is provided through a DB-15 type connector on the front panel, and supports digitizing of several different types of cameras. External programmable digital horizontal and vertical syncs are provided through the external DB-9 video interface connector so that users may

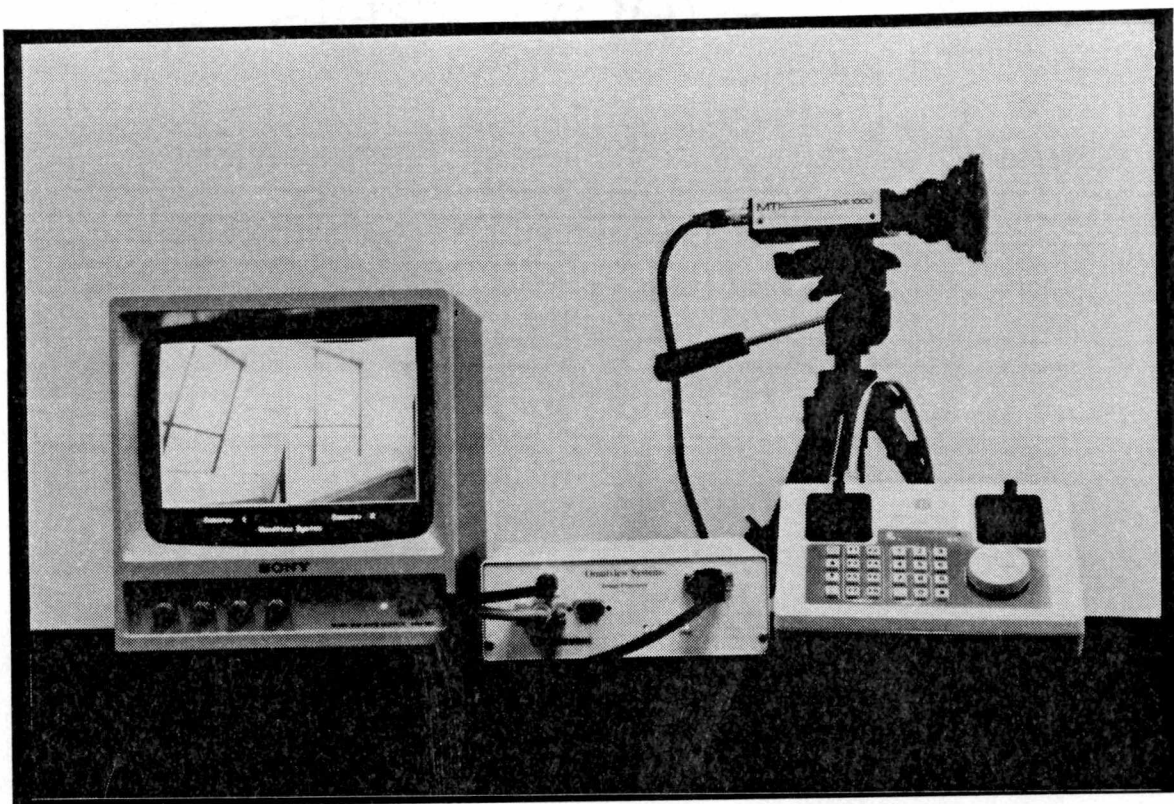


**Figure 1-1. Photograph of Prototype Omniview System**

generate standard or non-standard video syncs for different camera digitizing requirements. The output display is brought out through a 75-ohm BNC-type connector on the front panel. The generated display interfaces to most RS-170 compatible black-and-white monitors and provides NTSC standard black-and-white interlaced video or PAL-type video information. Non-interlaced video may also be generated by the output display port.

### **1.3 OVERVIEW OF CHAPTER CONTENTS**

The design and implementation of the Omniview camera system electronics is discussed in the remaining chapters. The development of the mapping algorithm that provides the fundamental equations on which the Omniview camera system is based is discussed in Chapter 2. The transformation equations that map the fisheye image into object plane coordinates is developed. Chapter 3 provides discussion of different image processing architecture approaches considered during the project development. Several technology enhancements led to the current Omniview image processor architecture and are discussed here. The design and implementation of the image process core and associated control logic are discussed in Chapter 4. The design and implementation of the host 32-bit

RISC processor core and remote interfaces is discussed in Chapter 5. The video processor board is presented in Chapter 6. Chapter 7 provides discussion of the operating system and software application development. Chapter 8 summarizes the results of the Omniview system development and implementation and discusses current and future development efforts.

## **1.4 SUMMARY**

The design and implementation of a prototype Omniview camera system is discussed in the following chapters. The Omniview system provides a number of enhancements over current viewing system approaches. Potential applications of the Omniview system include remote viewing in constrained environments, inspection, object tracking, image stabilization, and surveillance applications. Since the Omniview camera system can produce multiple images from a single fixed camera, it can replace several camera systems in a remote viewing application. The Omniview system architecture also has potential applications in a number of different image processing application areas.

## **CHAPTER 2**

# **DEVELOPMENT OF THE MAPPING ALGORITHM**

The Omniview camera approach makes use of the basic properties of a fisheye lens and the type of image it produces. Although wide-angle lenses have been in use in various applications such as surveillance and security and even robotics, most lenses have been limited to less than about a 110 degree field-of-view because of the amount of distortion present in the image. For very wide-angle lenses with field-of-view greater than about 110 degrees, the amount of distortion present becomes extremely noticeable. A fisheye lens provides even wider field-of-view images. The fisheye lens can provide up to a full hemispherical or 180 degree field-of-view.

The Omniview camera approach makes use of the special properties of a fisheye lens or ultra wide-angle lens equipped image system to generate an extremely flexible camera system for remote viewing. The special properties of the fisheye lens or wide-angle lens can be exploited to create a mapping transformation that corrects the barrel distortion present in the image to give a better representation of the image perspective. The mathematical transformations that provide the basis for the Omniview camera system are developed in this chapter.

### **2.1 FISHEYE LENS DESCRIPTION**

All lenses project circular images, but ordinarily, because of a relatively long focal length, the circle is large enough so that only a rectangular portion of the image is used. The shorter the focal length of a lens, the greater is its viewing angle. As the focal length is decreased, the amount of distortion in the image becomes greater. Some wide-angle lenses are manufactured to provide linear correction, however, as the focal length becomes shorter, it becomes more difficult and expensive to provide optical correction. A typical fisheye lens image is shown in Figure 2-1. The image distortion assumes a barrel-type

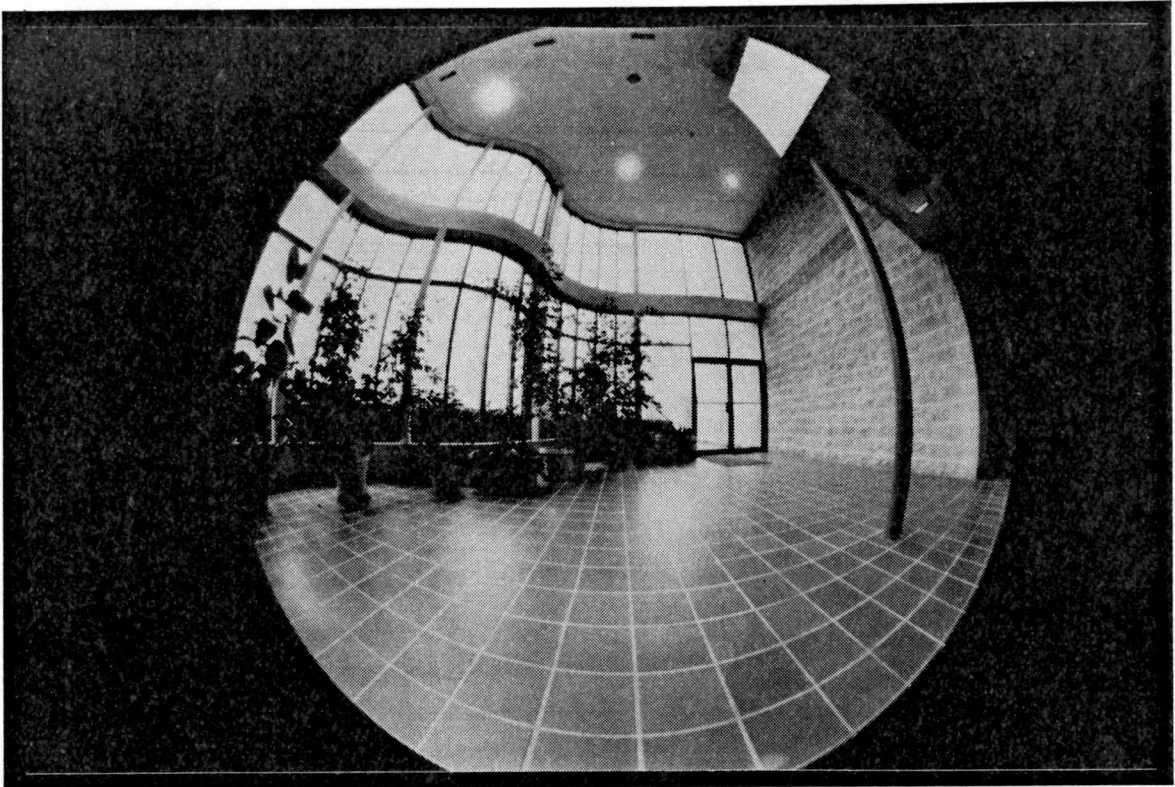


**Figure 2-1. Photograph of Typical Fisheye Image**

appearance in that objects off-center from the line-of-sight of the lens tend to be curved into circles. The amount of curvature present becomes greater the further off-axis the object is located. A fisheye lens provides a unique image in that the field-of-view is equal to 180 degrees or  $2\pi$  steradians and is usually referred to as a hemispherical lens because the image produced by the fisheye lens provides a picture of a complete hemispherical region mapped onto a circle. The mathematical development presented here assumes that a full 180 degree fisheye lens is being used for the image acquisition system. Smaller field-of-view lenses may also be used with the Omniview transformation approach.

### **Fisheye Lens Properties**

There are two basic properties and two basic postulates that describe the perfect fisheye lens system. The first property of a fisheye lens is that it encompasses a  $2\pi$  steradian or hemispherical field-of-view and produces a circular image. The fisheye lens encompasses a complete hemispherical field-of-view so that the image is symmetrical about the image center. The second property of the fisheye lens is that all objects in its field-of-view are in focus, i.e. the perfect fisheye lens has an infinite depth-of-field. In addition to

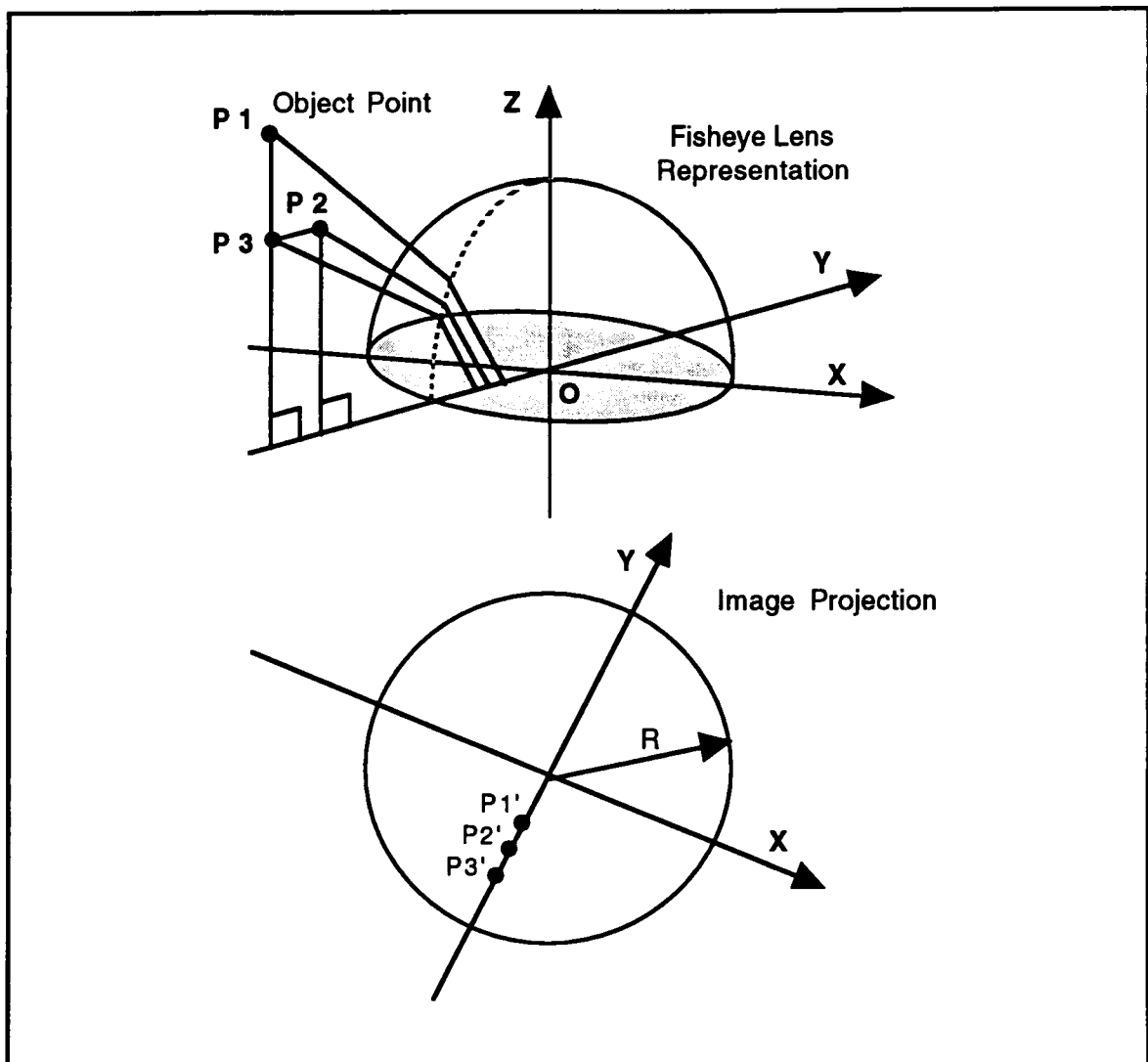


**Figure 2-2. Diagram of Azimuth Angle Invariability Property**

these two main properties, the two important postulates of the fisheye lens system are the azimuth angle invariability and the equidistant projection rule. These two properties are defined and represented graphically below. The following analysis uses these properties to formulate a set of mathematical transform equations for undistorting the image.

#### **Postulate 1: Azimuth angle Invariability**

The azimuth angle invariability property defines the relationship for object projection points that lie in the plane passing through the lens zenith axis. The azimuth angle invariability property is demonstrated graphically in Figure 2-2. For object points that lie in a plane that is perpendicular to the image plane and passes through the image

plane origin and containing the zenith axis, all such points are mapped as image points onto the line of intersection between the content plane and the image plane passing through the image center, i.e. along a radial line. The azimuth angle of the image points is therefore invariant to elevation and object distance changes within the perpendicular plane. In the figure, three objects points P1, P2, and P3 are projected through the fisheye lens onto the image plane. All three object points lie in the YZ plane. There is a change of height only in moving from object point P1 to object point P3, and a change of distance only in moving from object point P3 to object point P2. The projected image points, P1', P2', and P3', shown in the lower portion of the figure all lie on the radial line from the center of the image.

## **Postulate 2: Equidistant Projection Rule**

The equidistant projection rule relates the distance of the image point from the center of the image plane to the zenith angle of the object point. The equidistant projection rules states that the radial distance,  $r$ , from the image plane origin along the azimuth angle containing the projection of the object point is linearly proportional to the zenith angle  $\beta$ , where  $\beta$  is defined as the angle between a perpendicular line through the image plane origin and the line from the image plane origin to the object point. Thus the relationship:

$$r = k\beta \quad (2.1)$$

where  $k$  is a constant. The equidistant projection rule is demonstrated graphically in Figure 2-3. The distance from the image center to the object point is  $D$  and remains constant. The radius of the image circle is  $R$ . As the zenith angle,  $\beta$ , varies linearly from 0 to 90 degrees, the image point distance  $r$  varies linearly from zero to a maximum value  $R$  for a 180 degree lens.

## **2.2 TRANSFORMATION ALGORITHM DEVELOPMENT**

Using the properties and postulates of the fisheye lens as the foundation of the Omniview camera system, the mathematical transformation for obtaining a perspective corrected image can easily be determined. The following discussion develops the basic transformation algorithms in a simple and straightforward manner. The picture in Figure 2-4 shows the coordinate reference frames and axes assignments for the object plane and for

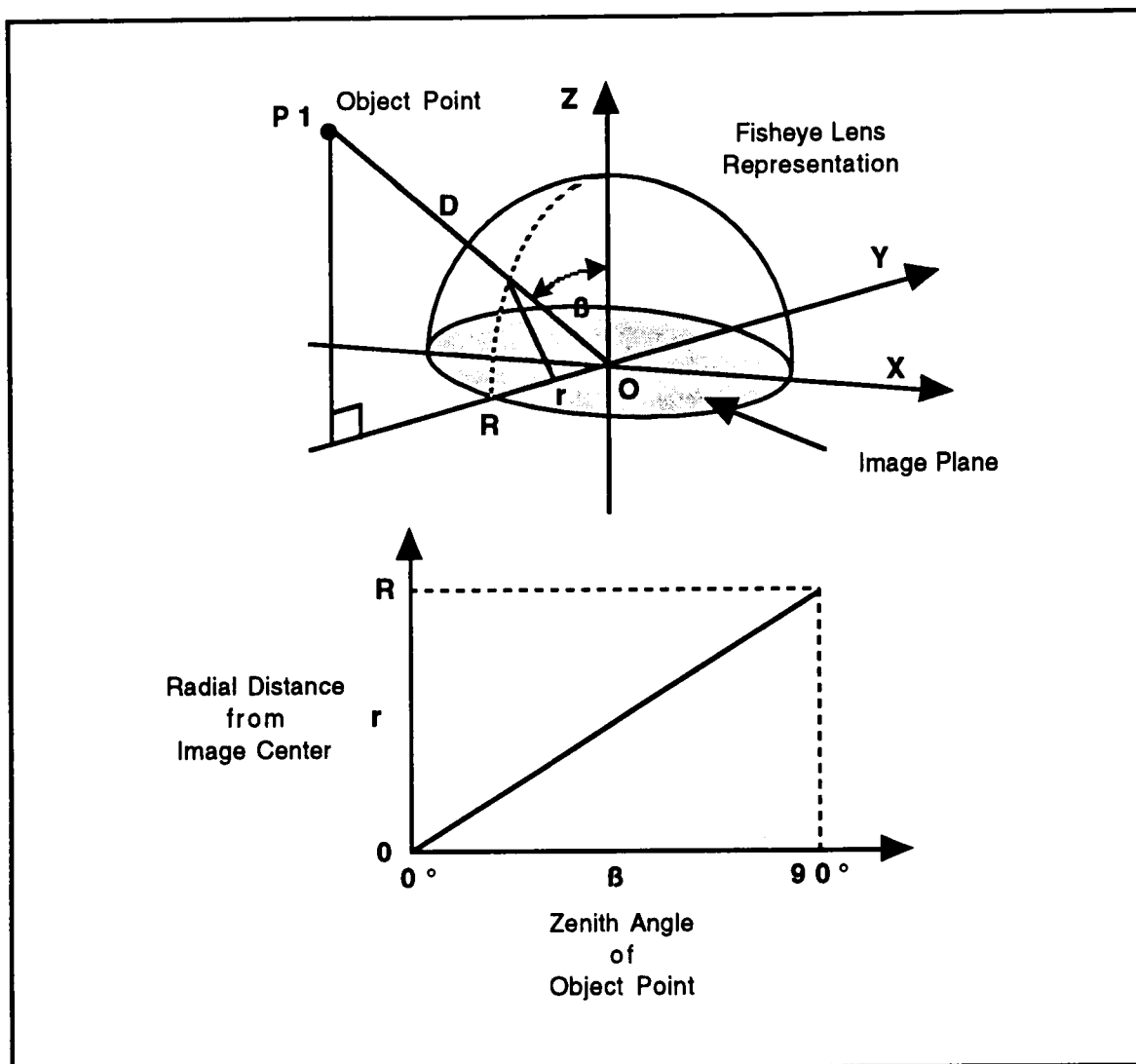


**Figure 2-3. Diagram of Equidistant Projection Rule**

the image plane. The coordinates  $u,v$  describe object points within the object plane. The coordinates  $x,y,z$  describe object points within the image coordinate frame-of-reference.

The object plane shown in Figure 2-4 represents a typical region-of-interest for which the mapping relationship is desired that determines the mapping of the object plane onto the image plane. The transformation provides the reverse mapping to generate the object plane to correct the proper perspective of the fisheye image. The direction-of-view vector,  $\text{DOV}[x,y,z]$ , determines the zenith and azimuth angles for mapping the object plane,  $\text{UV}$ , onto the image plane,  $\text{XY}$ . The object plane is defined to be perpendicular to the vector  $\text{DOV}[x,y,z]$ .

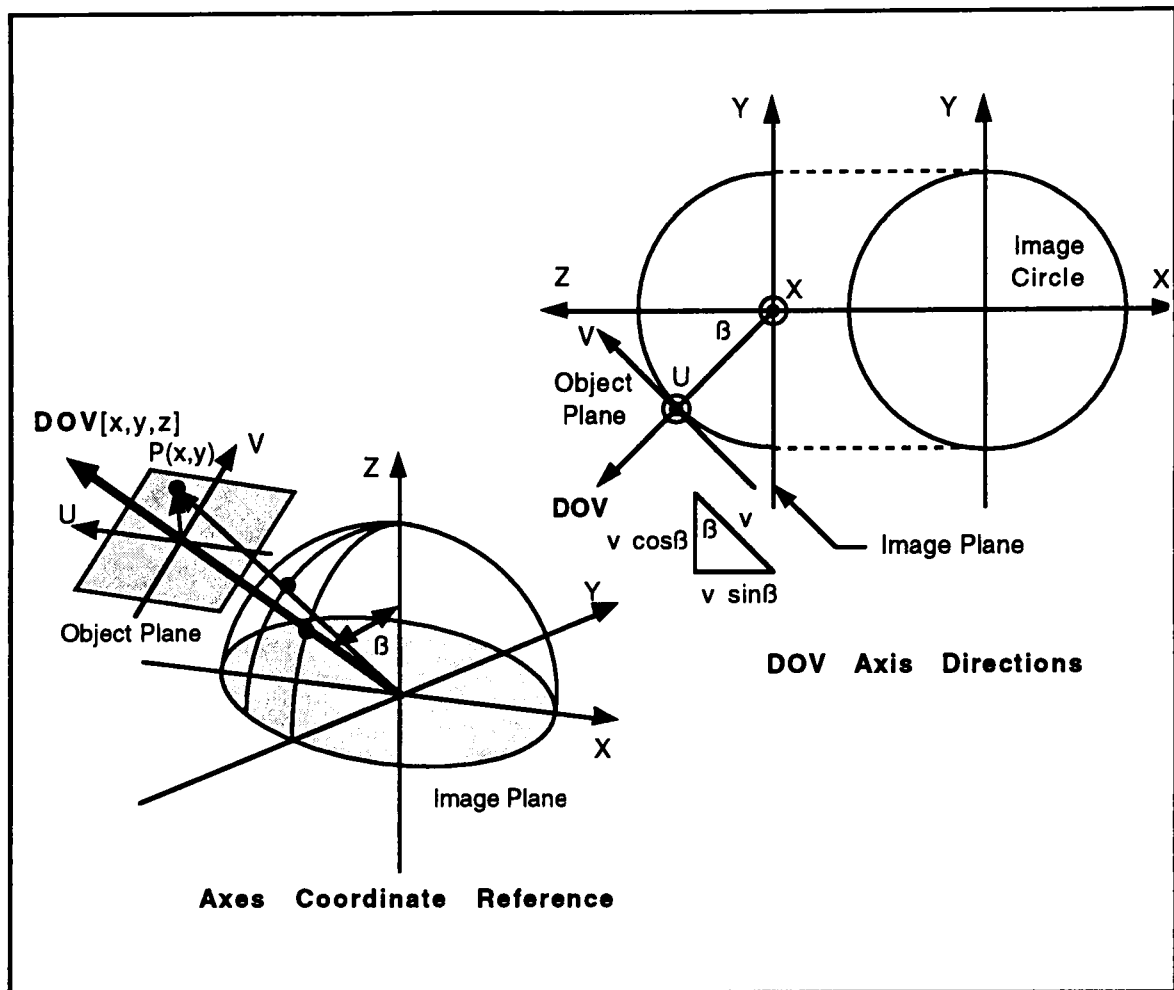


**Figure 2-4. Coordinate Reference Frame Representation**

The location of the origin of the object plane in terms of the image plane coordinates is given by:

$$\begin{aligned}
 x &= D \sin \beta \cos \vartheta \\
 y &= D \sin \beta \sin \vartheta \\
 z &= D \cos \beta
 \end{aligned}
 \tag{2.2}$$

where  $D$  is the scalar length from the image plane origin to the object plane origin,  $\beta$  is the zenith angle, and  $\vartheta$  is the azimuth angle in image plane spherical coordinates. The origin of the object plane is represented as a vector using the components given in equation 2.1 as:

$$\mathbf{DOV}[x,y,z] = [ D\sin\beta\cos\partial, D\sin\beta\sin\partial, D\cos\beta ] \quad (2.3)$$

The vector  $\mathbf{DOV}[x,y,z]$  is perpendicular to the object plane and its scalar magnitude  $D$  represents the distance from the image origin to the object plane. By aligning the  $YZ$  plane with the direction of action of  $\mathbf{DOV}[x,y,z]$ , the azimuth angle  $\partial$  becomes either 90 or 270 degrees and therefore the  $x$  component becomes 0 resulting in the  $\mathbf{DOV}[x,y,z]$  coordinates:

$$\mathbf{DOV}[x,y,z] = [ 0, -D\sin\beta, D\cos\beta ] \quad (2.4)$$

Referring again to Figure 2-4, the object point relative to the  $UV$  plane origin in coordinates relative to the origin of the image plane is given by the following:

$$\begin{aligned} x &= u \\ y &= v \cos\beta \\ z &= v \sin\beta \end{aligned} \quad (2.5)$$

Therefore, the coordinates of a point  $P(u,v)$  that lies in the object plane can be represented as a vector  $\mathbf{P}[x,y,z]$  in image plane coordinates:

$$\mathbf{P}[x,y,z] = [ u, v\cos\beta, v\sin\beta ] \quad (2.6)$$

where  $\mathbf{P}[x,y,z]$  describes the position of the object point in image coordinates relative to the origin of the  $UV$  plane. The object vector that describes the object point in image coordinates is then given by:

$$\mathbf{O}[x, y, z] = \mathbf{DOV}[x, y, z] + \mathbf{P}[x, y, z] \quad (2.7)$$

$$\mathbf{O}[x, y, z] = [u, v\cos\beta - D\sin\beta, v\sin\beta + D\cos\beta] \quad (2.8)$$

Projection onto a hemisphere of radius  $R$  attached to the image plane is determined by scaling the object vector  $\mathbf{O}[x,y,z]$  to produce a surface vector  $\mathbf{S}[x,y,z]$ :

$$\mathbf{S}[x,y,z] = \frac{R\mathbf{O}[x,y,z]}{|\mathbf{O}[x,y,z]|} \quad (2.9)$$

By substituting for the components of  $O[x,y,z]$  the vector  $S[x,y,z]$  describing the image point mapping onto the hemisphere becomes:

$$S[x,y,z] = \frac{RO[u, (v\cos\beta - D\sin\beta), (v\sin\beta + D\cos\beta)]}{\sqrt{u^2 + (v\cos\beta - D\sin\beta)^2 + (v\sin\beta + D\cos\beta)^2}} \quad (2.10)$$

The denominator in the last equation represents the length or absolute value of the vector  $O[x,y,z]$  and can be simplified through algebraic and trigonometric manipulation to give:

$$S[x,y,z] = \frac{RO[u, (v\cos\beta - D\sin\beta), (v\sin\beta + D\cos\beta)]}{\sqrt{u^2 + v^2 + D^2}} \quad (2.11)$$

From equation 2.11, the mapping onto the two-dimensional image plane can be obtained for both x and y as:

$$x = \frac{Ru}{\sqrt{u^2 + v^2 + D^2}} \quad (2.12)$$

$$y = \frac{R(v\cos\beta - D\sin\beta)}{\sqrt{u^2 + v^2 + D^2}} \quad (2.13)$$

Additionally, the image plane center to object plane distance  $D$  can be represented in terms of the fisheye image circle radius  $R$  by the relation:

$$D = mR \quad (2.14)$$

where  $m$  represents the scale factor in radial units  $R$  from the image plane origin to the object plane origin. Substituting equation 2.14 into equations 2.12 and 2.13 provides a means for obtaining an effective scaling operation or magnification which can be used to provide an equivalent zoom operation.

$$x = \frac{Ru}{\sqrt{u^2 + v^2 + m^2 R^2}} \quad (2.15)$$

$$y = \frac{R(v\cos\beta - mR\sin\beta)}{\sqrt{u^2 + v^2 + m^2 R^2}} \quad (2.16)$$

Using the equations for two-dimensional rotation-of-axes for both the UV object plane and the XY image plane axes, the last two equations can be further manipulated to provide a more general set of equations that provides for rotation within the image plane and rotation within the object plane.

$$x = \frac{R[ uA - vB + mR\sin\beta\sin\partial ]}{\sqrt{u^2 + v^2 + m^2 R^2}} \quad (2.17)$$

$$y = \frac{R[ uC - vD - mR\sin\beta\cos\partial ]}{\sqrt{u^2 + v^2 + m^2 R^2}} \quad (2.18)$$

where

$$\begin{aligned} A &= (\cos\partial\cos\partial - \sin\partial\sin\partial\cos\beta) \\ B &= (\sin\partial\cos\partial + \cos\partial\sin\partial\cos\beta) \\ C &= (\cos\partial\sin\partial + \sin\partial\cos\partial\cos\beta) \\ D &= (\sin\partial\sin\partial - \cos\partial\cos\partial\cos\beta) \end{aligned} \quad (2.19)$$

and where

$$\begin{aligned} R &= \text{radius of the image circle} \\ \beta &= \text{zenith angle} \\ \partial &= \text{Azimuth angle in image plane} \\ \emptyset &= \text{Object plane rotation angle} \\ m &= \text{Magnification} \\ u, v &= \text{object plane coordinates} \\ x, y &= \text{image plane coordinates} \end{aligned} \quad (2.20)$$

The two equations expressed in 2.17 and 2.18 provide a direct mapping from the UV space to the XY image space and provide the fundamental mathematical foundation for the omnidirectional viewing system with no moving parts. By knowing the desired zenith, azimuth, and object plane rotation angles and the magnification, the locations of x and y in the input image can be determined. This approach provides a means to transform an image from an input image memory buffer to an output image memory buffer exactly. Also, the

fish-eye image system is completely symmetrical about the zenith; therefore, the vector assignments and resulting signs of various components can be chosen to reflect the desired orientation of the object plane with respect to the image plane. This also allows the rotational orientation or azimuth of the camera to be compensated so that vertical orientations are appropriate. In addition, these postulates and mathematical equations can be modified for various lens elements as necessary for the desired field-of-view coverage in a given application.

### **2.3 SUMMARY**

The mathematical transformation equations for the Omniview camera system approach were developed using the properties of fish-eye lenses. The properties of the fish-eye lens were discussed in section 2.1. The mathematical transformation equations that provide the two-dimensional mapping function to generate a corrected object image from the fish-eye image were then developed in section 2.2. The mapping equations when applied to a fish-eye type image provide the ability to provide electronic pan, tilt, rotate, and zoom within a 180 degree hemispherical field-of-view. The completely mathematical approach for viewing provides a flexible and simple means to implement remote viewing camera systems where mechanical motion constraints or confined space entry requirements limit the size and motion range of conventional mechanical pan/tilt/zoom camera systems. Another advantage of the mathematical approach is the ability to provide multiple image viewing with independent pan/tilt/zoom operation from a single camera image or wide-angle lens equipped camera system. Since the image produced by the fish-eye lens is symmetrical, the rotational azimuth of the camera system can be accommodated to provide better viewing angles for the operator.

## **CHAPTER 3**

# **OMNIVIEW IMAGE PROCESSOR DEVELOPMENT**

The mathematical development in Chapter 2 provided the fundamental transformation algorithms that are used to implement the Omniview camera system. The goal of this chapter is to examine the different considerations of image processing architectures that were investigated during the development effort. Three basic approaches were recognized that provide the capability to implement the Omniview camera system. The advantages and disadvantages of each approach were considered and are briefly discussed.

### **3.1 PERFORMANCE CONSIDERATIONS**

The performance of the image processing system is an important issue for the implementation of the Omniview system. The desire is to provide the same performance as standard cameras. This involves image transformation frame rates on the order of 30 frame transformations-per-second. The video standard most prevalent is the Closed Circuit Television (CCTV) RS-170 standard. The RS-170 standard was designated for black-and-white and color baseband video transmission over coaxial cables. The RS-170 standard is much like the National Television Standards Committee (NTSC) baseband video transmission standard. The goal of the Omniview system is to provide an output image using the standard RS-170 format.

The RS-170 and NTSC video formats provide a goal for the frame rate performance and the pixel resolution to be accomplished by the Omniview system. The video formats are displayed in interlaced fields of 262.5 lines per field. The fields are separated into odd and even lines and are displayed at a rate of 60 fields-per-second. Two fields form a complete frame so that the frame rate is 30 Hz and consists of 525 lines per frame. The

minimum frame transformation goal for Omniview then is 30 transformed frames per second.

The resolution of the transformed image determines the pixel read/write performance goal. Since it is desired to support the NTSC format, the resolution of NTSC must be considered. The NTSC format provides a total of 525 lines per frame as discussed above. Although the number of lines per frame is fixed, the number of pixel elements per line is determined only by the bandwidth of the medium through which the image is to be transmitted. For the NTSC standard, the luminance bandwidth is fixed at about 4.5 MHz and therefore the maximum resolution is determined. The other factor involved is the aspect ratio of the displayed image. The NTSC standard was chosen to have an aspect ratio 4:3 or 4 horizontal pixels to 3 vertical pixels. The important outcome of this discussion is that the NTSC standard generally uses a 640 horizontal by 480 vertical pixel resolution for the active area of the image display. The non-active or non-displayed pixels are taken up during the horizontal and vertical blanking of the video image.

The Omniview transformation is symmetrical about the image center and therefore requires a square pixel format on both the input image and the output display. The NTSC output display is generated using a 640 by 480 square pixel display. Using the standard display rate of 30 frames-per-second, the total pixel transformation rate to support a 30 frame-per-second camera system is 9.2 million pixel-transforms-per-second. This requires that both equations for the X and Y functions to map the input pixel location as well as the U and V address to map the current output pixel must be calculated at the 9.2 million pixel-transform-per-second rate.

The Omniview transformation equations were developed previously and the total number of operations necessary to perform the image transformations can be approximated. Although different processing architectures have unique features to enhance processing speeds, the number of operations to implement the Omniview transformation can be estimated by considering each operation to be a fixed-point multiply or divide operation. Complex operations such as the square-root function needed in the denominator should be weighted since most hardware implementations do not support single cycle square-root calculations.

The X and Y equations each require eight operations to calculate the numerator function for a total of 16 arithmetic operations. Each denominator requires 7 simple arithmetic operations and a complex square-root operation. Finally, there are two precision divide operations which must be performed. The divide operation should also be considered as a complex operation since no general purpose microprocessor architecture supports single cycle divide instructions. Using the square-pixel NTSC format as the design goal, the number of calculations required to perform the transformations at the 30 frame-per-second rate can be calculated. There are a total of 23 simple operations, and a total of 3 complex operations. If we use a standard shift-and-add architecture and assume 16-bit results, the complex operations should be given a weight of 16. This means that the complex operations require an equivalent of 48 simple operations. The final total operations required to implement a fixed-point implementation is 71 simple operations. Assuming that a simple instruction requires a single clock cycle to perform, the Omniview transformation would require a processor capable of generating 654 million arithmetic instructions-per-second. When the read and write operations and the address generation are included, the processing rate increases to about 672 million operations-per-second. This estimate does not include the generation of the coefficients since these values need to be generated only once for each viewing direction change.

### **3.2 GENERAL PURPOSE APPROACHES**

The Omniview transformation equations can be implemented using many different processing systems. The transformation algorithms can be implemented using software on high performance processing boards or on general purpose personal computer platforms. Some of the advantages and disadvantages are discussed in this section.

#### **General Purpose CPU Approach**

A general purpose CPU or personal computer can be used to provide the first level in the implementation of the Omniview camera system. The general purpose CPU or personal computer is attractive mainly because it provides the capability to offer a complete operator interface as well as control of other functions such as lighting and video switching equipment along with the Omniview transformations. The general purpose computer has the advantage of needing only the implementation of the software algorithms to accomplish the transformation. The video input and display interface hardware can be purchased off-

the-shelf for most personal computers. Very little hardware design expertise would be needed to implement a complete camera system.

The disadvantages associated with a general purpose computer implementation are performance and cost. Most personal computers even today operate at a maximum simple instruction rate of about 25 million instructions-per-second for only simple operations such as add and subtract. As was discussed previously, the instruction rate for the Omniview transforms would require at least a 680 MIP personal computer. The personal computer implementation is also extremely costly. In order to capture and display video images, a separate video processor board is required which generally costs several hundred dollars. The bandwidth of the personal computer backplane implementation is also an issue. If several systems installed in the personal computer backplane need to share the backplane bus, the arbitration logic will slow down transfers significantly.

The general purpose computer approach is still viable for implementation of the Omniview transformations though. The general purpose computer approach can provide the ability to implement remote viewing of static images. For example, a pipe inspection system can be implemented where the basic need might be to examine the interior for deformations, rusting, or some other condition.

### **High Performance Parallel Processing**

High performance parallel processing systems which can be applied to the implementation of the Omniview transformations are becoming available. The advantage to using these high performance parallel processors is again the flexibility gained in programming and manipulation of the image data. Special convolution or morphological processing functions can be implemented in the software programming. The major disadvantage to using these systems is their prohibitive cost.

### **Image Warping Systems**

The industry supports several image processing architectures implemented on dedicated boards and backplanes. An example is the Datacube Inc.'s MaxVideo image warping and transformation board. The advantages of using the special image processing hardware are the flexibility and the programmability of the software and hardware operations. The disadvantage in the operation is that the Omniview transformations need

both a numerator and denominator implementation. This would require multiple image warping boards. Another disadvantage is that current industry image warping systems require use of a backplane system and a separate host computer system to control the operation. The VME system contains the largest number of image processing systems, however, the VME implementation would require a large backplane and card cage and a large electronic enclosure. Since these implementations also are fairly expensive other approaches were considered.

### **3.3 XY ADDRESS LUT IMPLEMENTATION**

One interesting architecture approach for the implementation of the Omniview camera system was proposed by the author in the initial proposal and project efforts and led to the successful funding of the project by NASA. The transformation approach makes use of the special architecture of video memory devices or VRAMs which were very new components. The VRAM device is a combination of DRAM memory technology but contains the ability to provide dual-port access to the memory array.

A block diagram of the architecture approach is shown in Figure 3-1. The XY transformation buffer approach contains a camera genlock digitizer interface and a frame buffer to capture the fisheye image. The frame buffer indicated in the block diagram suggests a 2048 by 2048 pixel input image capture buffer. The frame buffer implementation could support either black-and-white or color camera system implementations.

The output display interface consists of a separate output display frame buffer and associated display generation logic. The display logic would provide standard NTSC interlaced display at a pixel resolution of 640 horizontal by 480 vertical resolution.

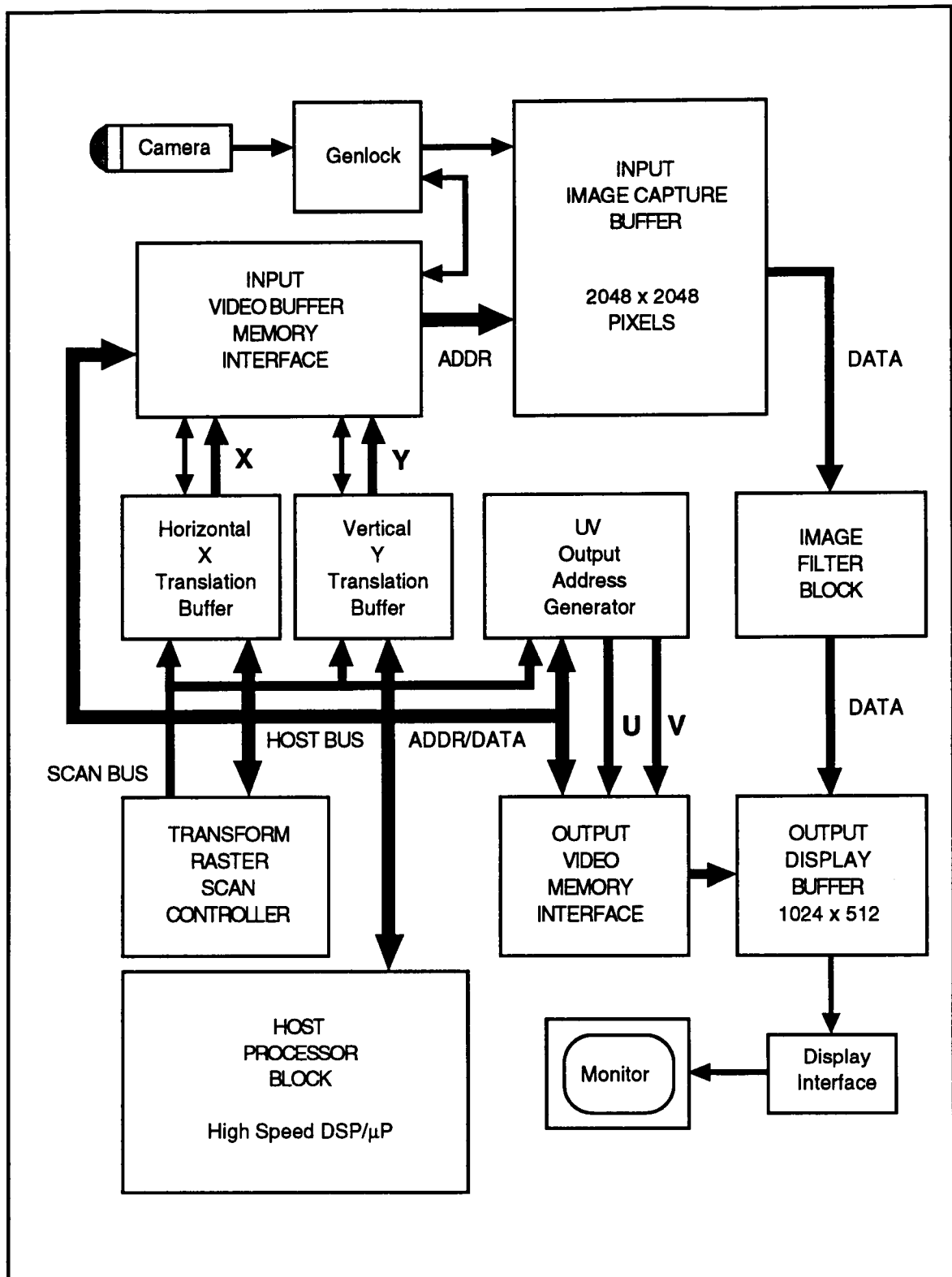


**Figure 3-1. XY Address LUT Block Diagram Approach**

## **XY LUT Operation**

The XY LUT (look-up-table) approach is attractive for implementation of the Omniview cameras system because it offers the capability to perform transformations at the 30 frame-per-second rate. The address generation is performed by the VRAM XY translation buffers and the UV address buffers.

As shown in the functional block diagram, Figure 3-1, the input image capture and storage buffer would consist of up to 2048 by 2048 pixels. The width of the pixel data in bits is not of concern because the image transformation approach is independent of the pixel depth. The 2048 by 2048 input image buffer is desired because it supports the highest resolution cameras currently available.

The XY and UV translation buffers are implemented using 1-Megabit VRAM devices. These devices are organized as 512 rows by 512 columns by 4 bits of data. Each of the X and Y translation buffers consists of three of the VRAM devices in order to form a complete 12-bit address. Since the input image buffer is only 2048 by 2048 elements requiring an 11-bit address for each dimension, the extra bits of address could be used to mask the read/write transfers. For example, one of the mask bits would provide the capability to generate windows where transforms would be allowed. Whenever the scan operation is outside a valid window, the transfer operation would be ignored and the address would be skipped to the next generated address. Another mask bit could be used to select different filtering modes.

## **VRAM Memory Implementation**

The XY translation buffer approach makes use of a unique memory technology to accomplish the real-time transformations. The X and Y address generation blocks each use video memory devices or VRAMs. The VRAM device technology was fairly new when the initial architecture for the implementation of the Omniview system was considered. A simplified functional block diagram of a 1 Megabit VRAM internal architecture is shown in Figure 3-2. The VRAM is a dual-port dynamic random access memory with both a DRAM parallel data access port and a serial (SAM) access port.

The DRAM array for the 1 Megabit VRAM devices is organized as 512 rows by 512 columns with 4-bit data. The DRAM array is accessed using a multiplexed address.

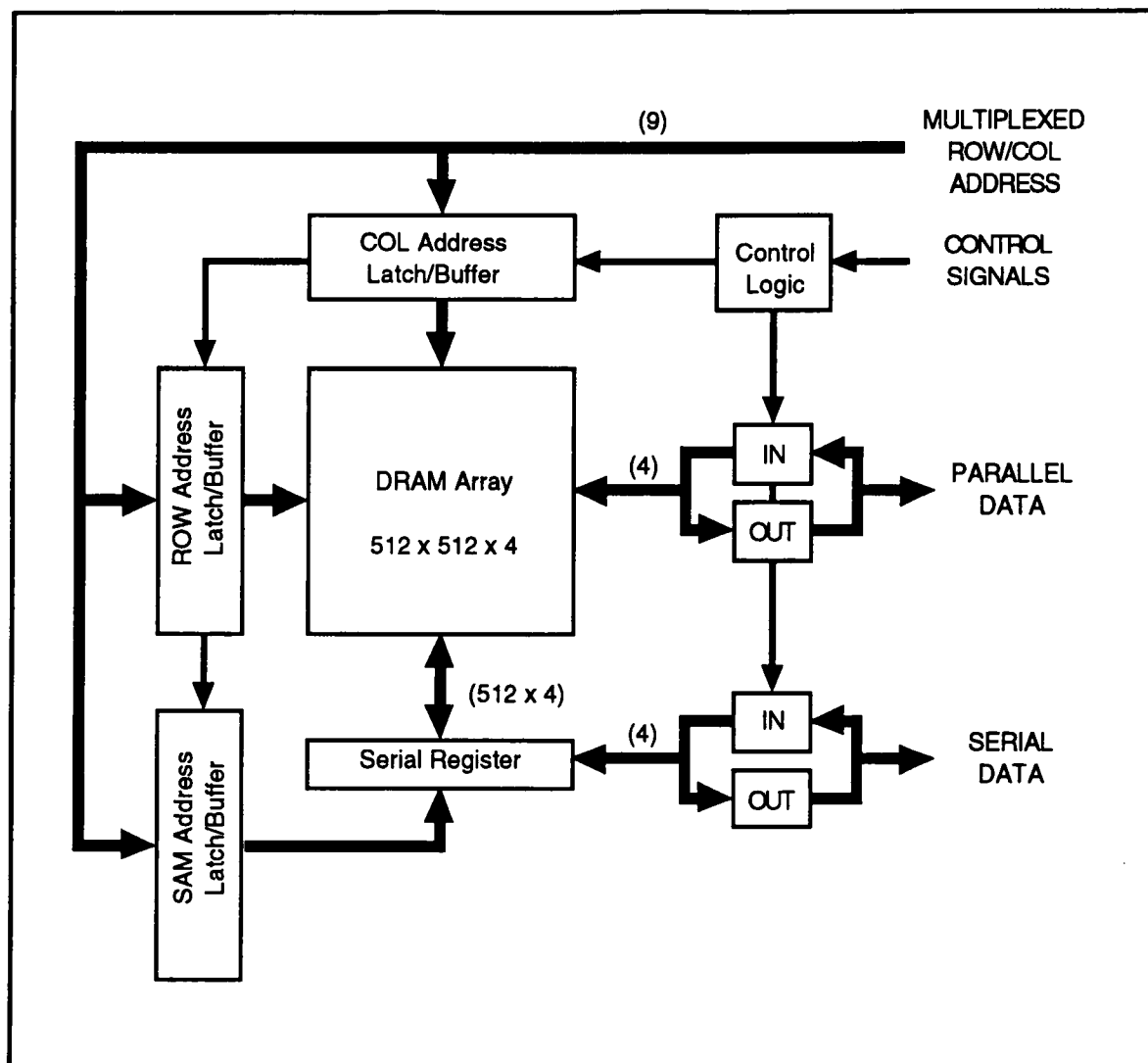


**Figure 3-2. VRAM Functional Block Diagram**

The address is latched internally to generate first the row address and then the column address.

The SAM port consists of bi-directional data buffers and a 512 by 4-bit serial access register. The SAM register connects to the 512 columns of the memory array and data may be either read from the memory array or written into the memory from the SAM register 512 words at a time. Each port of the VRAM may be accessed asynchronously and independently except when data is being transferred to or from the SAM register. This synchronous transfer capability makes the VRAM ideal for the implementation of the Omniview XY LUT transformation approach.

## XY Translation Buffer Operation

The operation of the translation buffer approach is designed to be self-synchronous and to generate the image transforms at the 30 frame-per-second rate. A brief description of the operation is described. The X and Y translation buffers provide the 12-bit address to access a particular pixel in the image buffer. In conjunction with the XY translation buffers, a UV generator is implemented to address the output display buffer. The addresses to access the output UV address could be implemented using direct single stepped addresses; however, the VRAM buffer approach is again attractive because it offers the capability to generate windows in the output image buffer. Each window would be synchronized with the window transforms in the input XY translation buffer in terms of physical scanned address but would have different offset address. For instance, the input image buffer might be implemented in two's complement format to make the calculation easier while the output buffer might be in the standard zero-based address format - (0,0) to access the upper left pixel.

The Raster Scan Generator (RSG) provides the scan and synchronization of the addresses to perform the actual transform. The RSG operates in the same manner as a video graphics display controller. An address clock is divided to generate the lines of the data scan while the system clock generates the addresses. A complete line of addresses in a row of the X and Y translation buffer VRAMs is transferred into the SAM memory of the VRAMs. Once the DRAM to SAM transfer operation has been completed, the SAM register contains a complete line of transform addresses. Each address is then serially scanned out of the SAM. This addresses are serially scanned until a complete line has been generated. The RSG then performs a transfer operation to load a new line of data into the SAM and the next line of transforms is ready to be scanned. The RSG also generates U and V addresses in the same manner and maintains synchronization with the X and Y addresses except for some pipeline delay to accommodate the filter processing.

A graphical representaton of the XY LUT transformation operation is shown in Figure 3-3. As shown in Figure 3-3, the composite XY address generated by the combination of the X and Y translation buffers is shown with a cross-hatched window. The cross-hatched window indicates the valid addresses which would generated pixel transforms. The RSG generator provides a direct mapping from the input image buffer to the output image buffer in a raster scan generation fashion. Although the addresses are

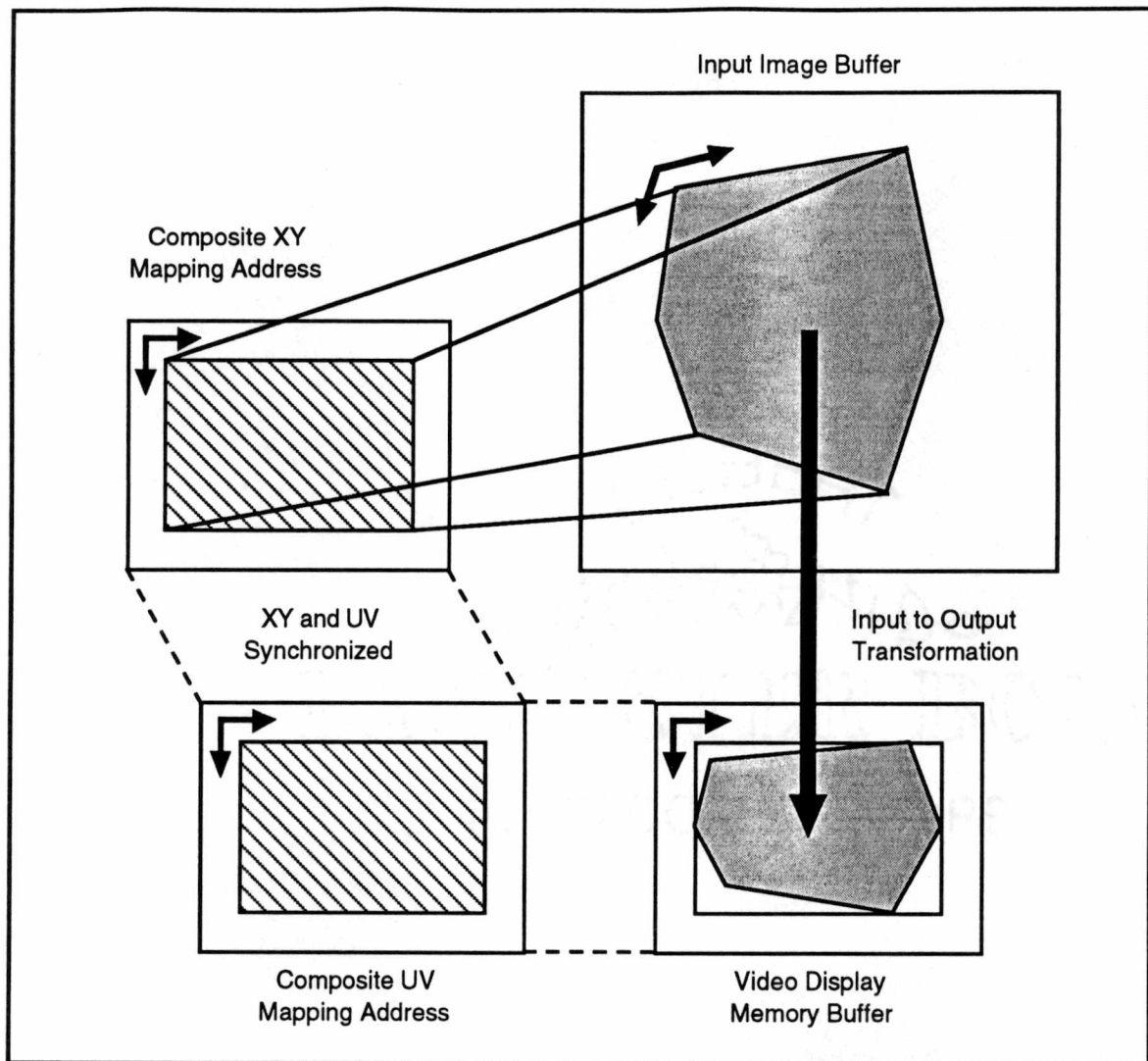


**Figure 3-3. XY LUT Transform Graphical Representation**

scanned in a line-by-line fashion, the addresses generated can access pixels in both the input and output memory buffers in any manner. A single address can be stored in the XY translation buffers to magnify a single pixel to any size in the output display. This approach provides a great deal of flexibility for viewing the output images. Several windows can be created each providing different viewing situations with independent pan, tilt, rotation, and magnification.

The main disadvantage with the XY translation buffer approach is that the host processor must generate all of the LUT addresses for a particular transform. The time for a single image pan might take on the order of one or two seconds for even the fastest general

purpose microprocessor. The advantage though is that once a transform has been completed, the image transforms operate at real-time rates. The security and surveillance industries would be able to use this approach since the general viewing applications involve coverage of particular areas and do not normally need dynamic scan. The ability to provide multiple windows and viewing directions would be an advantage. One window view could be used to provide a wide-angle view while a second window could be used to focus on an area of interest within the overall field-of-view. Another advantage to the XY translation buffer approach is that the cost of implementation would be significantly reduced when compared to dedicated high speed parallel processing components or image warping systems currently available.

### **3.4 DEDICATED HARDWARE APPROACH**

The final approach considered in the design and implementation of the Omniview camera system is the design of a dedicated hardware image processor. A functional block diagram of the dedicated hardware processor Omniview system is shown in Figure 3-4. The system includes all of the interface components as discussed previously including a host processor to provide computerized remote controlled interfaces and hand-held remote display control. The image transform processors are implemented using high speed arithmetic components.

The desire for implementing a dedicated Omniview image transformation processor is to provide the capability to perform real-time image manipulation and image stabilization operation. There are many applications that could use the Omniview camera system for which real-time operation is necessary. One example is in the application of remote viewing cameras for remotely guided unmanned aerial vehicles or UAVs. The Omniview system provides several important advantages for the UAV systems currently being designed. Since the Omniview system provides pan/tilt/zoom (PTZ) viewing over the entire captured image, the UAV system can be equipped with only a single wide-angle high resolution camera. The entire image is transmitted to the ground where the Omniview system processes the image to provide the remote viewing operation. Since all of the information is present in the input image, it is not necessary to transmit data in closed-loop-control mode back to the remote UAV for PTZ operation. The image stabilization and pan-tilt-zoom operation can be accomplished on the ground at the control site which may be located several miles away from the UAV. Another advantage to the real-time Omniview

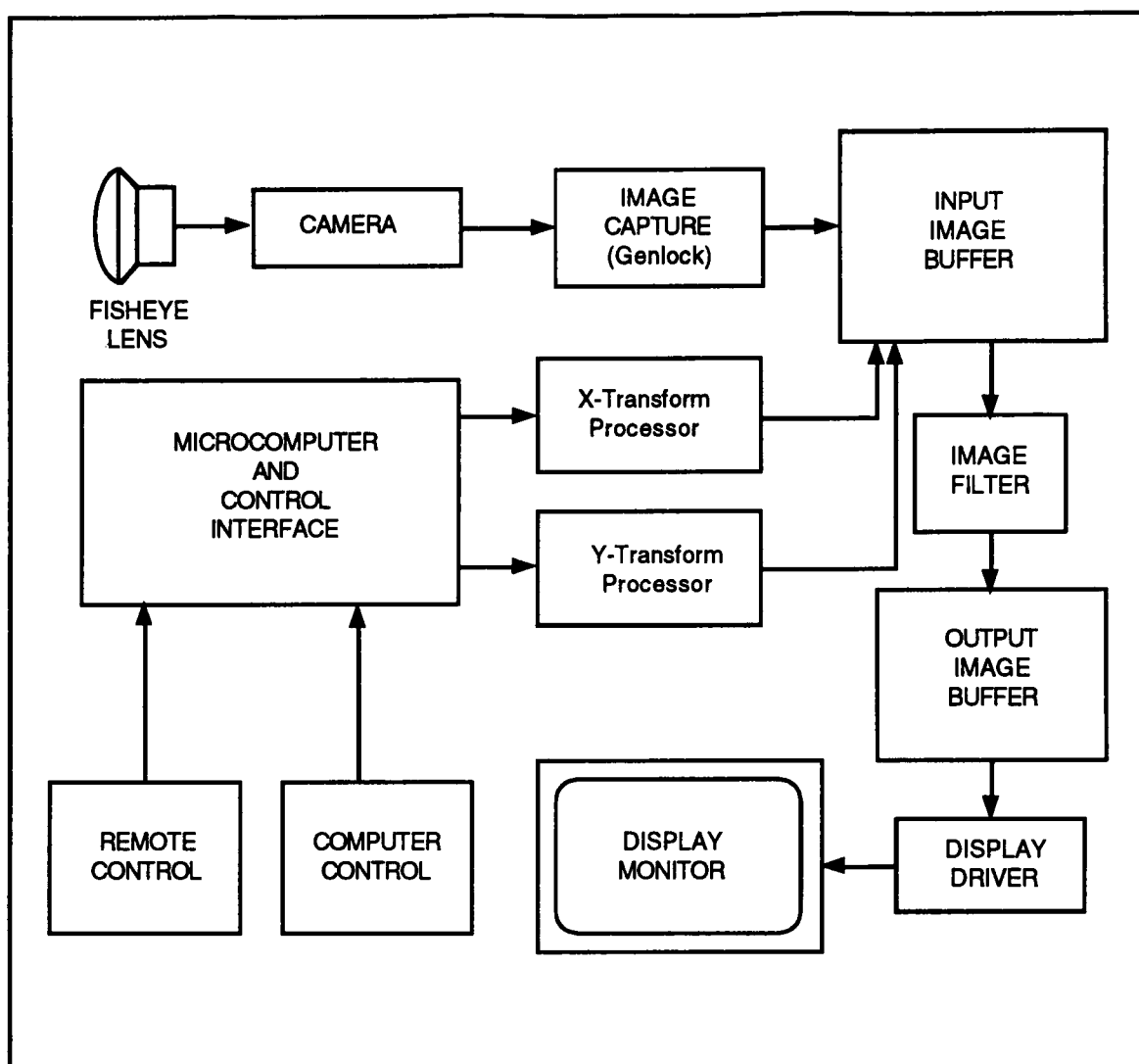


**Figure 3-4. Omniview Functional Block Diagram**

approach is that the same video image transmitted from the UAV can be spilt and processed by separate Omniview systems to provide a great deal of flexibility. One Omniview system can be dedicated to piloting and control of the UAV aircraft while secondary Omniview systems provide viewing and reconnaissance information all from the same remote camera. The other added advantage is the reduction in weight, volume, and power requirements that would be necessary for mechanical PTZ systems.

The Omniview transformation equations are repeated below in equations 3.1 and 3.2. The main observation to be made about the implementation of the equations is the symmetrical nature of the X and Y mapping equations. Both the X and Y equations are

similar except for the coefficients. There are four coefficients A,B,C, and D which are outlined in equations 3.3 in addition to two SIN coefficients and a COS coefficient used in the equations. The viewing parameters are given in equations 3.4 and determine the direction-of-view, rotation, and magnification. The symmetrical nature of the X and Y functions are reasonable since the approach relies on a symmetrical lens and input image.

The equations can be further broken down into an X and Y numerator equation and a denominator equation. The X and Y numerator equations are presented again in equations 3.5 and 3.6. Only one denominator equation is needed for both transforms and is presented again in equation 3.7.

$$x = \frac{R[ uA - vB + mR\sin\beta\sin\partial ]}{\sqrt{u^2 + v^2 + m^2 R^2}} \quad (3.1)$$

$$y = \frac{R[ uC - vD - mR\sin\beta\cos\partial ]}{\sqrt{u^2 + v^2 + m^2 R^2}} \quad (3.2)$$

where

$$\begin{aligned} A &= (\cos\partial\cos\beta - \sin\partial\sin\beta\cos\beta) \\ B &= (\sin\partial\cos\beta + \cos\partial\sin\beta\cos\beta) \\ C &= (\cos\partial\sin\beta + \sin\partial\cos\beta\cos\beta) \\ D &= (\sin\partial\sin\beta - \cos\partial\cos\beta\cos\beta) \end{aligned} \quad (3.3)$$

and where

$$\begin{aligned} R &= \text{radius of the image circle} \\ \beta &= \text{zenith angle} \\ \partial &= \text{Azimuth angle in image plane} \\ \emptyset &= \text{Object plane rotation angle} \\ m &= \text{Magnification} \\ u, v &= \text{object plane coordinates} \\ x, y &= \text{image plane coordinates} \end{aligned} \quad (3.4)$$

Numerator Equations:

$$x = R[ uA - vB + mR\sin\beta\sin\partial ] \quad (3.5)$$

$$y = R[ uC - vD - mR\sin\beta\cos\partial ] \quad (3.6)$$

Denominator Equation:

$$x = \sqrt{u^2 + v^2 + m^2 R^2} \quad (3.7)$$

The denominator equation was developed previously and shown to be independent of rotation. The denominator term depends only on the size of the output object plane, the magnification, and the radius of the image. Except for the square-root function, the denominator term can be easily implemented with discrete arithmetic components using multipliers and adders.

The equations 3.5, 3.6, and 3.7 suggest a numerator and denominator approach to the implementation of the Omniview image transform processor. A functional block diagram of the numerator/denominator approach is shown in Figure 3-5. The implementation consists of a host processor block, a UV processor, separate X and Y processors, a Root-Sum-of-Squares (RSS) generator, and high speed digital dividers. One important point to note about the Omniview equations is that the addresses are assumed to be two's complement. The U and V scan addresses are positive binary numbers and are used to access the display memory buffer where the transformed image will be written. The actual values of the UV scan address will be different from the two's complement UV address in the denominator. The UV processor is necessary to generate the output image. The UV values which locate the image windows in the output display are not the same as the UV object plane values. The X and Y numerators are each implemented in a separate block and provide the numerator portion of the equations. The denominator portion of the transformation equations are implemented by a single RSS generator. The final X and Y values are generated using two high speed divider components. The output of the dividers represents the image transformation addresses that would be used to index into the input image buffer. The X and Y address from the divider outputs are in two's complement format and needs further processing to interface to the image frame buffer. The numerator and denominator terms can be functionally separate but must be synchronized. The output of the high speed divider is the address to be presented to the input image buffer.

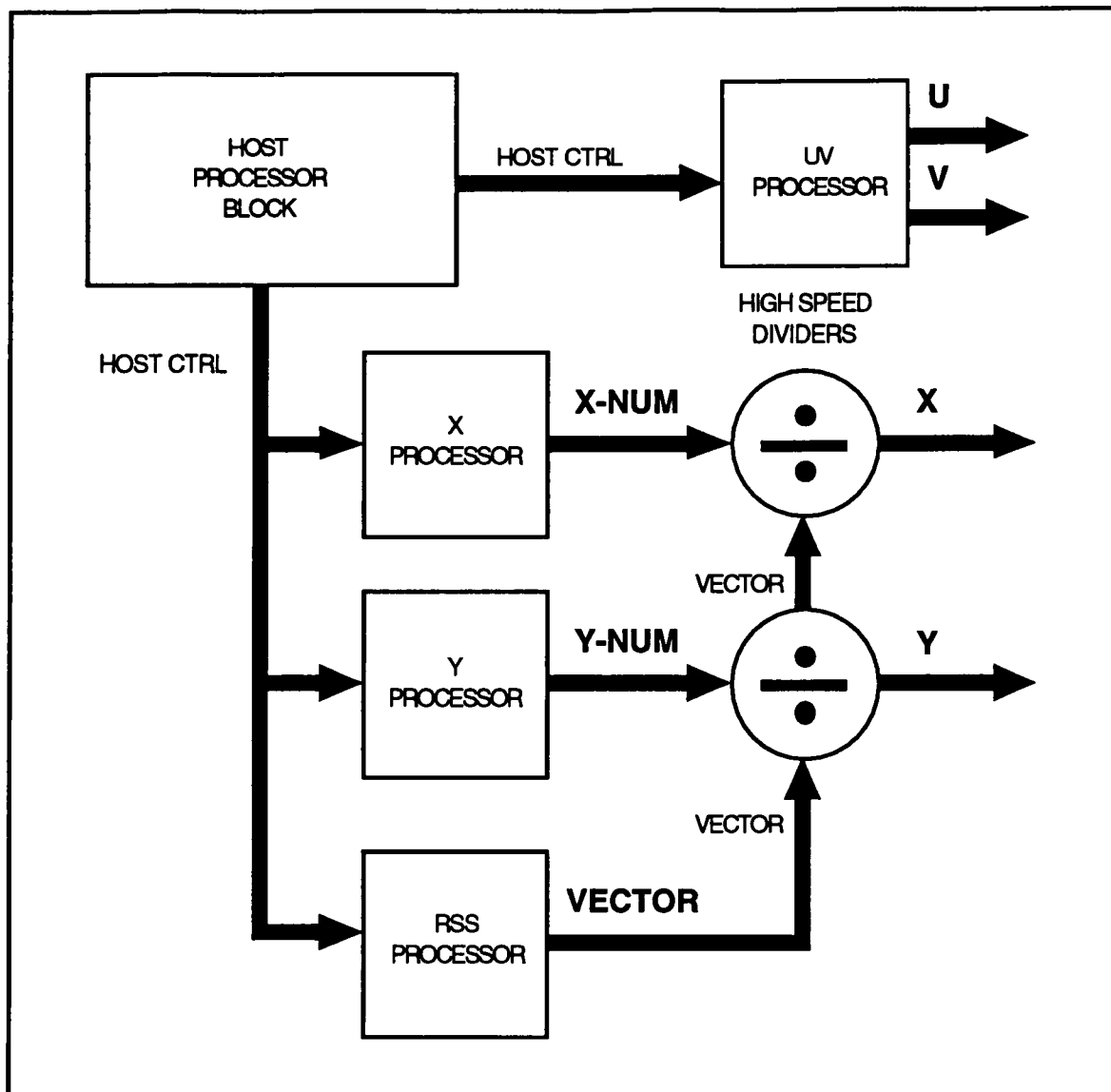


**Figure 3-5. Dedicated Numerator/Denominator Approach**

### **Vector RSS Generator Discussion**

A functional block diagram of one possible implementation of the denominator or RSS processor is shown in Figure 3-6. The RSS processor generates the square-root of the sum of the squares of the U,V and mR values. The U and V addresses in the denominator are two's complement numbers. A separate UV generator is needed for the denominator since the U and V addresses defined in the denominator equation are different from the U and V scan address which actually access the output display memory buffer.

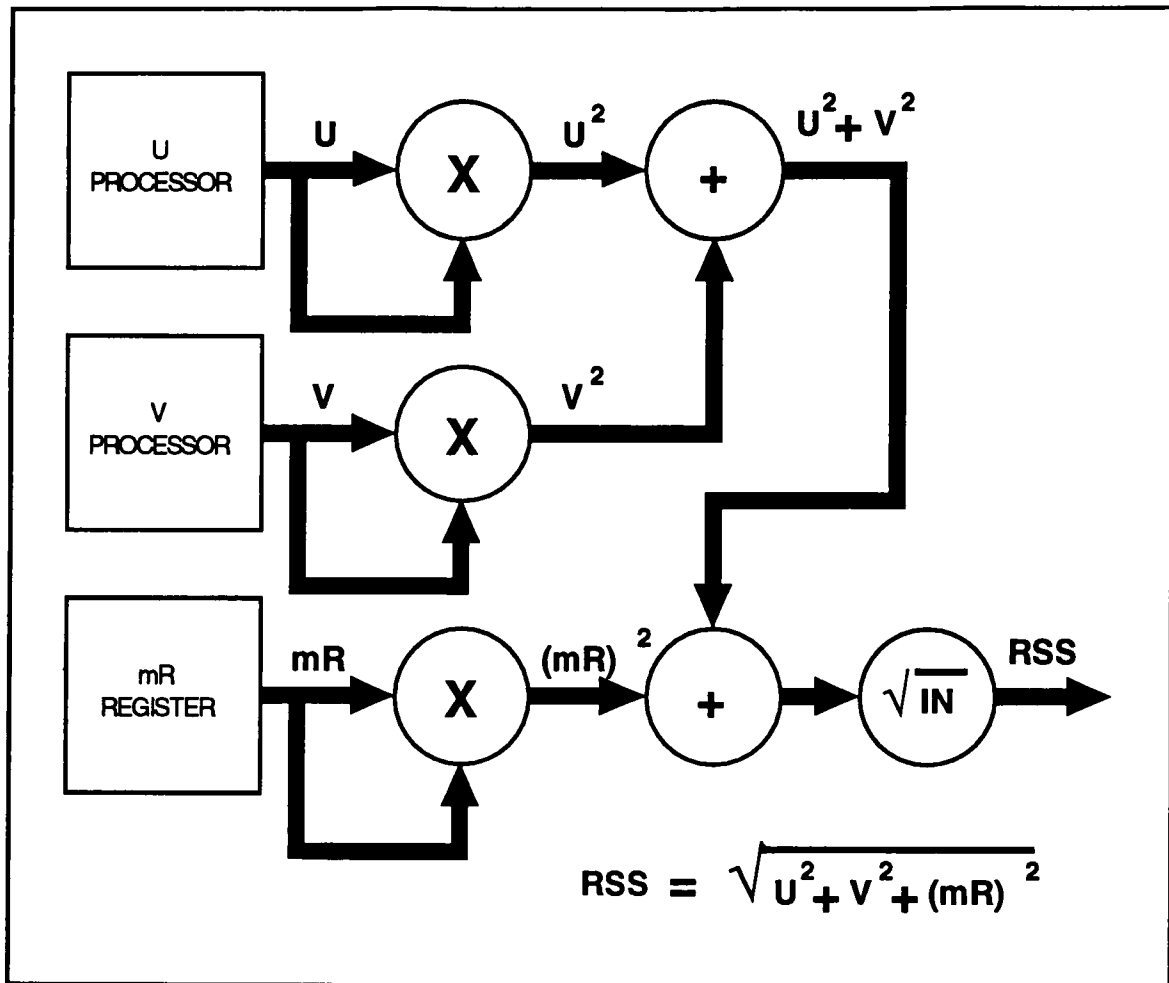


**Figure 3-6. RSS Generator Functional Block Diagram**

Each component is squared using a high speed multiplier. The squaring function can possibly be implemented in an ASIC or gate array architecture. The outputs of the three squaring blocks are summed using two adders. The square-root of the output of the adders is performed using a special square-root function.

### 3.5 SUMMARY

The design and implementation of the Omniview camera system was considered in this chapter. The Omniview image processor transformation hardware is designed to provide NTSC standard image display and to support real-time 30 frame-per-second transformations. An analysis of the performance suggests that a 680 MIP 32-bit microprocessor would be needed to implement the Omniview camera transforms to provide a 640 by 480 pixel resolution display system.

Several different approaches were considered for the implementation of the Omniview system. All of the approaches have merit for particular applications. Two approaches were discussed that have the capability to provide real-time transformations. The first approach is the XY LUT translation architecture. The XY LUT approach makes use of the unique architecture of VRAM memory devices to provide synchronous real-time transformations with the ability to support multiple windows with different viewing parameters. The advantage of using the XY LUT approach is the simplicity of the implementation and the flexibility in programming. The main disadvantage with the XY LUT approach is that it is slower than dedicated hardware when the viewing parameters such as the direction-of-view or magnification are changed. For static viewing directions or windowed viewing applications, the XY LUT approach represents a viable alternative to the dedicated hardware approach. The second approach considered is the implementation of dedicated arithmetic processing hardware to implement the real-time transformations. The advantage of the dedicated hardware approach is the ability to perform real-time image manipulations such as image rotation and image stabilization. The primary disadvantages to the dedicated hardware is the cost and complexity of the implementation.

The dedicated hardware implementation was chosen as the first implementation of the Omniview system and is discussed in the following chapters. The Omniview system consists of three main processing boards. The image processor board implements the hardware execution of the Omniview transformations algorithms. The host processor provides the system support and interface and the calculation of the image transform parameters and programming. The video processor is responsible for the capture, filtering, and display of the video images.

## **CHAPTER 4**

# **OMNIVIEW IMAGE PROCESSOR DESIGN**

Several design approaches for implementing the Omniview image transformation processor were presented in Chapter 3. Each approach provides advantages and disadvantages and must consider the needs of the user's application. The initial demonstration of the Omniview camera system concept was accomplished using a personal computer to perform the image transformations in software. The discussion in this chapter provides an overview of the design and implementation of the dedicated hardware implementation of the Omniview image transformation processor. The dedicated hardware processor approach offers the advantage of real-time image transformations to support 30 frame-per-second video camera systems.

### **4.1 IMAGE PROCESSOR ARCHITECTURE**

The Omniview image processor was designed to function as a stand-alone image processor board. The goal of the design is to provide an image processor board that can be embedded in different Omniview camera system applications. The Omniview image processor board was designed and fabricated on a single printed circuit board. The design approach allows the image processor to be integrated into different host processor configurations.

The image processor design consist of a numerator and a denominator processor to generate the numerator and denominator terms of the transform equations that were developed in Chapter 2. A functional block diagram of the image processor is shown in Figure 4-1. One of the goals in the design of the image processor core was to be able to use the image processor in different applications. This goal was met by using a single printed circuit board to contain the image processor with interface connectors. The image processor electronics is fabricated on a single 6-inch by 8-inch printed circuit board.

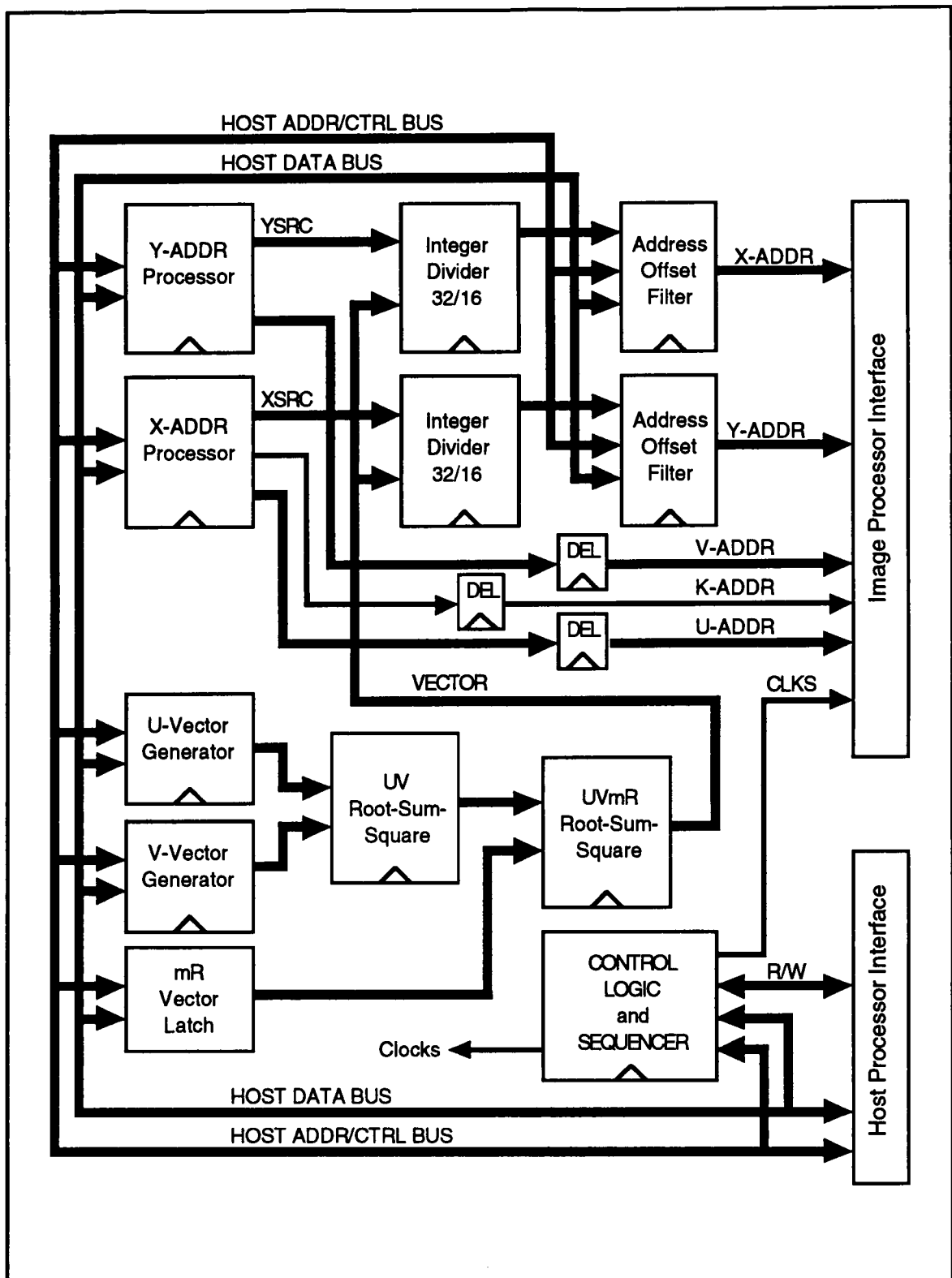


Figure 4-1. Image Processor Functional Block Diagram

A high speed integer divider and address filter section performs the final stage of the transformation equation calculation. The image processor also contains some pipeline processing support components and specialized control logic design. A pipeline sequencer controls the sequencing and clocking of the numerator and denominator processors. A read/write control section provides an asynchronous interface to the video frame buffer memory. Two 56-pin interface connectors provide the physical interface to the host processor and video frame buffers. One connector contains all of the host processor address, data, and control logic interface signals. The other connector provides all of the video frame buffer addressing and control interface signals. Each of the major functional blocks and the specialized components that are used in the implementation are discussed in this chapter.

## **4.2 NUMERATOR PROCESSOR**

The Omniview image processor design makes use of a separate numerator and denominator processor to accomplish the high speed transformations. The numerator processor was implemented using a special address generator device from TRW LSI Products Inc. designated as the TMC2302. The TMC2302 is an Advanced Image Manipulation Sequencer (AIMS) and provides all of the processing power necessary in order to implement the numerator processing functions for the image processor.

### **AIMS Processor Description**

The TMC2302 AIMS processor is a self-sequencing large-scale CMOS integrated circuit which provides extremely flexible address generation capabilities. A functional block diagram of the AIMS is shown in Figure 4-2. The AIMS is a high-speed address generator that supports image resampling, rotation, scaling, warping, and filtering of images at a speed of up to 40 MHz in current technology.

The AIMS is designed to act as a co-processor in an image resampling implementation and requires that a host processor provide programming and configuration by loading the transformation registers internal to the device. The AIMS is also designed to provide two-dimensional address generation when using two of the devices. One device is used to generate the X address dimension while second device is used to generate the Y dimension in an application.

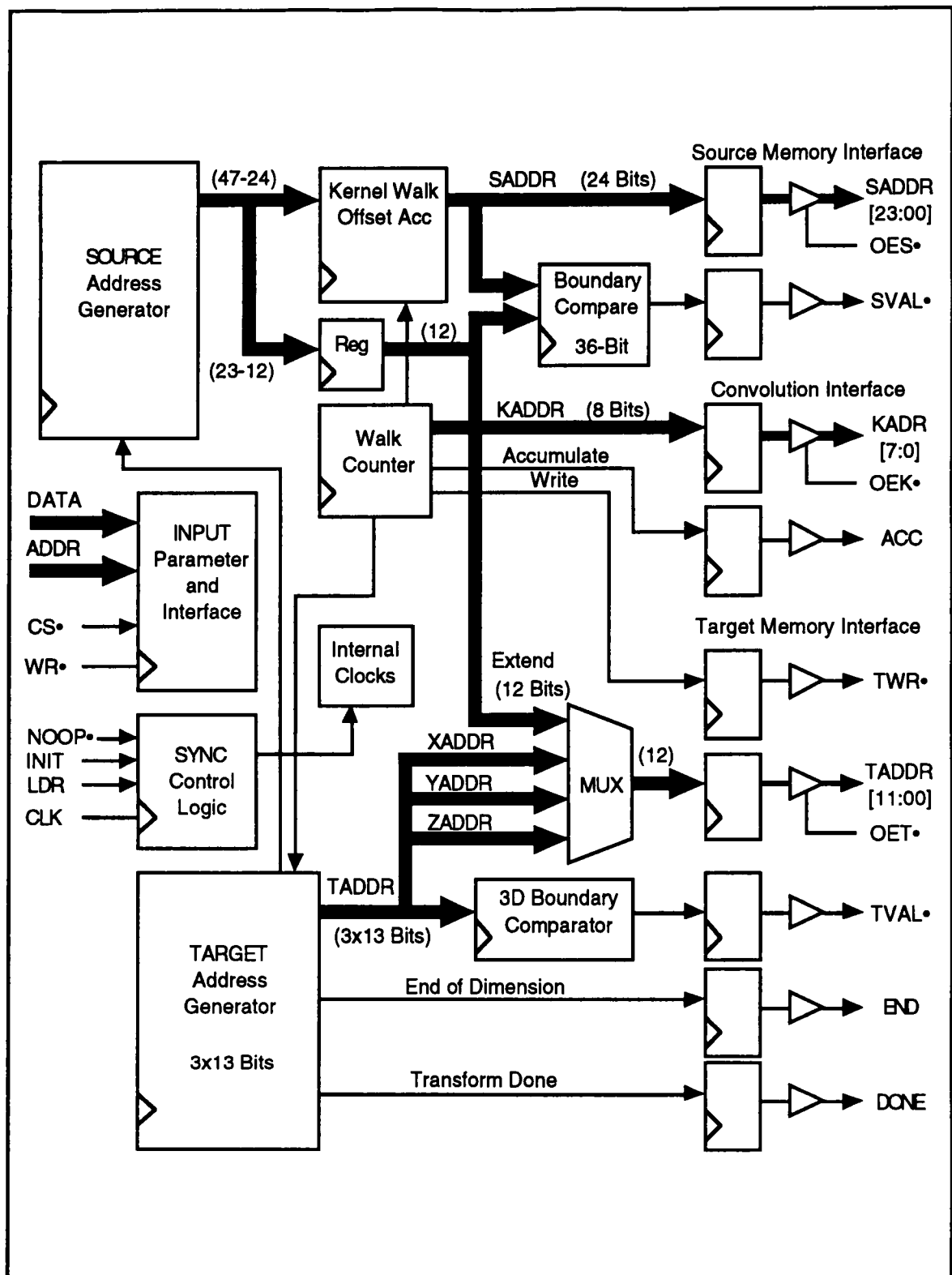


Figure 4-2. AIMS Functional Block Diagram

The AIMS contains five main functional blocks internally - the source address generator, the target address generator, the interpolator walk counter, the input parameter registers, and the control logic. The AIMS provides both input source X and Y address generation and output target U and V address generation. The AIMS internal source address generator uses 48-bit precision coefficients and can implement up to a third order two-dimensional address transformation polynomial. Each device in the X and Y implementation of the address generator can provide support for a 16-term polynomial equation of the following form:

$$X(U,V) = a + bU + cU^2 + dU^3 + eV + fVU + gVU^2 + hVU^3 + iV^2 + jV^2U + kV^2U^2 + lV^2U^3 + mV^3 + nV^3U + oV^3U^2 + pV^3U^3 \quad (4.1)$$

The second dimension or Y address is generated by a second AIMS which generates the similar equation:

$$Y(U,V) = a + bV + cV^2 + dV^3 + eU + fUV + gUV^2 + hUV^3 + iU^2 + jU^2V + kU^2V^2 + lU^2V^3 + mU^3 + nU^3V + oU^3V^2 + pU^3V^3 \quad (4.2)$$

Each of the coefficients  $a$  through  $p$  of the X and Y equations are programmable and are calculated to a precision of 48-bits. Both X and Y devices therefore require thirty-two 48-bit coefficients to determine one transformation. The AIMS uses three levels of internal 48-bit accumulators to calculate the polynomial by using forward-difference accumulation. As can be seen, the AIMS is capable of a wide range of image or address manipulation applications. Although the polynomial coefficients are maintained internally to 48-bit precision, only the upper 24 bits of the calculated address are brought out externally during normal operation. The transformation equation for the AIMS device can also be written in the following forms:

$$X(U,V) = a + eV + iV^2 + mV^3 + (b + fV + jV^2 + nV^3)U + (c + gV + kV^2 + oV^3)U^2 + (d + hV + lV^2 + pV^3)U^3 \quad (4.3)$$

and

$$Y(U,V) = a + eU + iU^2 + mU^3 + (b + fU + jU^2 + nU^3)V + (c + gU + kU^2 + oU^3)V^2 + (d + hU + lU^2 + pU^3)V^3 \quad (4.4)$$

Using the forms for the polynomial equations in equations 4.3 and 4.4, the polynomial coefficients can be represented in arithmetic terms as shown in Table 4-1. As the polynomial form indicates, the source address is a polynomial function of two dimensions of the target address. Each of the sixteen terms of the equation is of the form of a mixed partial difference equation and can be represented in the following form:

$$\frac{d^{m+n}}{dU^m dV^n} \quad (4.5)$$

where  $m$  and  $n$  are integers.

Each of the coefficients may be treated as a mixed partial difference of order  $m$  and  $n$ . The simplest term is  $X_0$  and is a zeroth order function of the target addresses. The  $X_0$  term specifies the initial source address point corresponding to the upper left point in the object plane for the  $X$  address generator device. The same term,  $Y_0$ , is generated for the  $Y$  address generator device.

The next and simplest higher order terms,  $(m,n) = (1,0)$  and  $(m,n) = (0,1)$ , are the first order non-mixed differences and represent relative scales of the source addresses with respect to the target image. These terms represent how large a step in source image space corresponds to a unit step in the target image space.

$$\frac{dX}{dU} \quad (4.6)$$

$$\frac{dY}{dV} \quad (4.7)$$

The first-order cross-terms are then represented as follows:

$$\frac{dX}{dV} \quad (4.8)$$

$$\frac{dY}{dU} \quad (4.9)$$

If the cross-terms in equations 4.8 and 4.9 are zero, the first-order terms in equation 4.6 and 4.7 provide a straight linear scaling operation in each dimension. The first-order cross terms provide an additional source displacement perpendicular to the unit displacements in the target space.

**Table 4-1. AIMS Polynomial Coefficients**

| Name | Difference Term         | Polynomial Coefficient |
|------|-------------------------|------------------------|
| A    | $X_0$                   | a                      |
| B    | $\frac{dX}{dU}$         | b+c+d                  |
| C    | $\frac{d^2X}{dU^2}$     | 2c+6d                  |
| D    | $\frac{d^3X}{dU^3}$     | 6d                     |
| E    | $\frac{dX}{dV}$         | e+i+m                  |
| F    | $\frac{d^2X}{dUdV}$     | f+g+h+j+k+l+n+o+p      |
| G    | $\frac{d^3X}{dU^2dV}$   | 2(g+k+o) + 6(h+l+p)    |
| H    | $\frac{d^4X}{dU^3dV}$   | 6(h+l+p)               |
| I    | $\frac{d^2X}{dV^2}$     | 2i+6m                  |
| J    | $\frac{d^3X}{dUdV^2}$   | 2(j+k+l) + 6(n+o+p)    |
| K    | $\frac{d^4X}{dU^2dV^2}$ | 4k+12l+12o+36p         |
| L    | $\frac{d^5X}{dU^3dV^2}$ | 12l+36p                |
| M    | $\frac{d^3X}{dV^3}$     | 6m                     |
| N    | $\frac{d^4X}{dUdV^3}$   | 6(n+o+p)               |
| O    | $\frac{d^5X}{dU^2dV^3}$ | 12o+36p                |
| P    | $\frac{d^6X}{dU^3dV^3}$ | 36p                    |

The cross terms therefore can add rotation or shear of the source image. The higher order terms can be further defined up to third order for the AIMS. For instance, the AIMS will support source image warping with the following non-mixed terms:

$$\frac{d^3X}{dU^3} \quad (4.10)$$

$$\frac{d^3Y}{dV^3} \quad (4.11)$$

and the following mixed terms:

$$\frac{d^6X}{dU^3dV^3} \quad (4.12)$$

$$\frac{d^6Y}{dU^3dV^3} \quad (4.13)$$

A list of all of the difference terms and their corresponding polynomial coefficients are listed in Table 4-1. The list shown in Table 4-1 shows all the difference terms for the X address generator only. The Y address generator provides the same terms with respect to the Y source address. Although the warping actions of the higher order terms becomes difficult to describe, all of the terms behave essentially as partial differences of various orders. For example, the second-order term

$$\frac{d^2}{dU^2} \quad (4.14)$$

provides a horizontal scaling operation which increases as the transform progresses across a row. The effect produced is a horizontal quadratic warping of the source image. All non-mixed terms of the form

$$\frac{d^mX}{dU^m} \quad (4.15)$$

$$\frac{d^nY}{dV^n} \quad (4.16)$$

cause a non-linear scaling or stretching of the source image and never rotation or shearing.

The U and V target memory addresses are generated internally for each X and Y AIMS device while only one dimension is brought out externally. The X device provides the U target addresses while the Y device provides the V target addresses. If the U and V addresses are generated externally to the AIMS devices, the U and V target address pins can be used by the AIMS device to generate an additional 12 bits of source address to provide a 36-bit precision source address.

The Omniview system uses two of the AIMS devices to generate 24-bit X and Y source memory addresses for the numerator terms and 12-bit U and V target addresses. The X and Y source addresses provide the numerator terms to 24-bit precision. The U and V outputs provide the output target memory window addresses and allow a target memory buffer implementation of up to 4096 by 4096 pixels. The Omniview implementation provides an output display resolution of 640 by 480 pixels and therefore needs only 10-bit horizontal and 9-bit vertical address resolution for the U and V addresses.

The AIMS device is programmed through a write-only register interface. The AIMS supports the capability to implement several different output display window transforms for the Omniview system through the parameter pre-load register interface. The input parameter register interface is buffered to allow the transform coefficients to be loaded for a new transform while the current parameter set is being used for a transformation. When the AIMS device reaches the end of the current transform dimension, it automatically loads the next set of parameters from the parameter pre-load registers. The Omniview host system must therefore insure that each set of parameters be loaded for the new transform before the current transform is finished.

The AIMS implementation also provides the capability to filter an image using a two-dimensional transform walk accumulator. When an external multiplier-accumulator is used with the AIMS implementation, the system can provide two-dimensional filter kernels with up to 256 coefficients. The Omniview system uses an 8-bit multiplier-accumulator (MAC) with an 8-bit kernel coefficient memory to provide the capability to generate either nearest neighbor, bilinear, or cubic convolution filtering. The main disadvantage with the resampling 2D filter is that it requires a much higher bandwidth and faster memory. For example, a bilinear filter implementation would require the transform to read 4 pixels for each output pixel which would slow the transform frame rate by a factor of four for a fixed bandwidth memory implementation.

### 4.3 DENOMINATOR PROCESSOR

The denominator processor design provides the equivalent root-sum-square operation as discussed in Chapter 3. The denominator processor or RSS processor provides the square root of the sum of the squares of the U and V target addresses and the mR vector. The equation implemented by the RSS processor is:

$$RSS = \sqrt{U^2 + V^2 + (mR)^2} \quad (4.17)$$

The RSS processor was implemented by using a pair of Actel FPGA devices to generate the U and V vector scan addresses and a special arithmetic component that implements the equivalent of the square-root of the sum of the square of its inputs. The following sections provides a functional overview of the RSS implementation.

#### UV Vector Generator Implementation

The vector generator block provides the equivalent of the U and V coordinates for the output image plane. The U and V addresses are implemented using two identical Actel 68-pin FPGAs. A functional block diagram of the UV-vector generator is shown in Figure 4-3. The vector generator device contains six internal registers organized as three sets of parameter registers. The three parameters are the start, end, and increment values for the particular dimension and are implemented as two's complement 16-bit integers. The start value determines the 16-bit starting point of the particular dimension. The end point register contains the value of the last vector value in the 1D dimension scan of the vector generator. The increment register provides the incremental step value between each generated vector. The 16-bit vector generator is implemented using a 16-bit accumulator, a 16-bit counter register and a 16-bit two's complement adder. During each step of the U or V scan, the current increment is added to the current count and the result serves as the next input to the counter register. The counter register also has a 2-to-1 multiplexor to select either the next count value or the next starting value under control of the enable and control logic. When the vector scan generator reaches the end of the current dimensional scan, the counter register or accumulator is reset to the new starting vector value as determined by the View input.

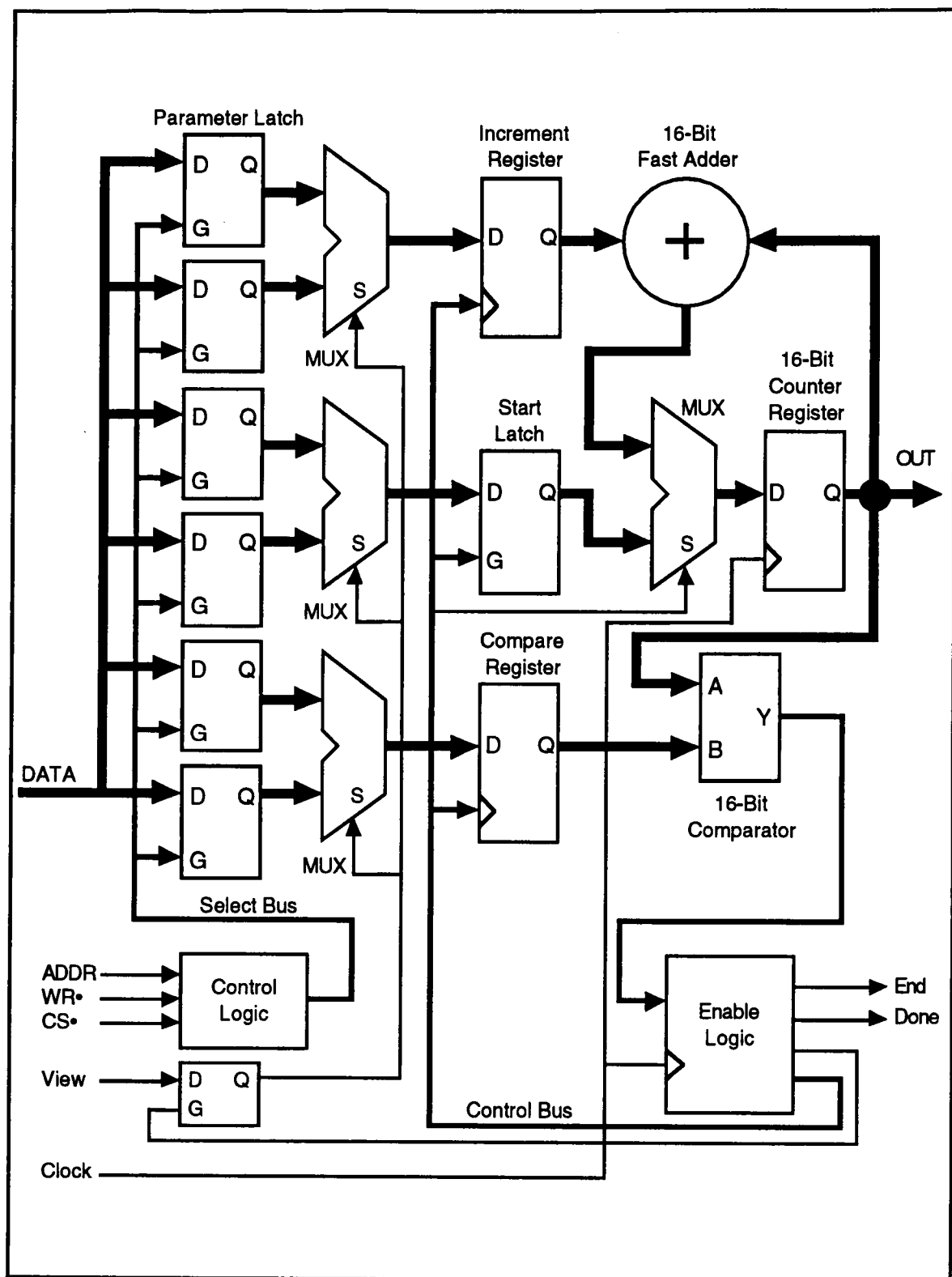


**Figure 4-3. UV Vector Generator Functional Block Diagram**

A particular parameter register set is selected by the View input. The View input controls a 16-bit 2-to-1 multiplexor. The outputs from the multiplexors are latched into three separate parameter registers. These registers are clocked at the end of each dimension scan to begin each new horizontal U vector or vertical V vector for the current image. For instance, when the U vector device reaches the end of the current line in the output, the 16-bit accumulator and counter register is loaded with the new start value as selected by the start latch and latched View input. During each vector clock, the increment value is accumulated with the current counter register value. The end of the current dimension is indicated by the 16-bit comparator block which compares the magnitude of the current counter value and the end point register latch. The control and enable logic synchronizes the loading of the parameter latch registers and the counter register logic. The control logic decodes the CS\*, WR\*, and ADDR signals from the host processor interface to control writes to the internal parameter registers. External control signals are brought out to allow two vector generator devices to be connected together in order to generate the two-dimensional object plane coordinates in a raster scan fashion. One device implements the U dimension while the other provides the V dimension. Each time the U-vector generator reaches the end of its horizontal scan, the V-vector generator is enabled and generates the next vertical value. When both the U-vector and V-vector generators reach the end of their respective dimensions both devices reset to their starting values as determined by the current View. The View signal is internally registered in order to allow the View signal to change before the end of the UV scan. The new view is only changed when both devices reach the last point of the current two-dimensional UV scan.

### **Actel FPGA Designs Considerations**

The Omniview design uses several Field-Programmable Gate Arrays (FPGAs) in the implementation. FPGAs combine the density and flexibility of a gate-array architecture with the convenience, ease-of-use, and time-to-market benefits of a user-programmable standard product. Although mask-programmable gate arrays offer the architecture flexibility and efficiency to integrate thousands of gates, they require long and costly development cycles and high initial engineering costs. On the other hand, standard PALs and PLAs offer convenience of re-programmability, high speed, and relatively low cost; however, they do not support the densities and capabilities of FPGAs. FPGAs are also programmed by the end user or designer. As a result, the designer retains control of the

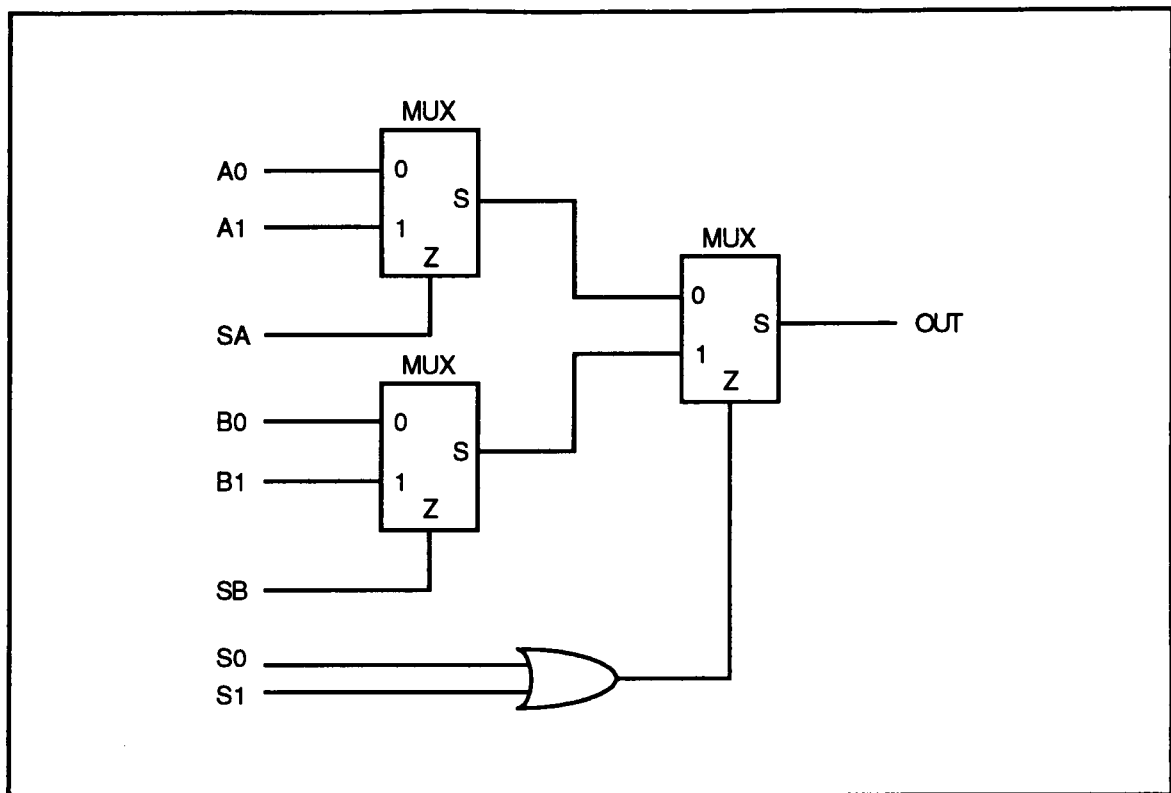


**Figure 4-4. Actel ACT™ 1 Logic Module Architecture**

entire development process avoiding the long, costly design cycles associated with custom ASIC device designs.

The Actel ACT-1 series of FPGAs were chosen because of the integration density, speed, and flexibility of the devices and design environment. The Actel FPGA architecture utilizes a single programmable logic module to provide multiple logic implementations. The Actel ACT-1 family of FPGAs uses only one type of logic module. The ACT-1 module, shown in Figure 4-4, has eight inputs and a single output. It is composed of three two-to-one multiplexers, with an OR gate on the last stage's two-to-one multiplexor select input. Using this module configuration, various macros can be implemented by using an appropriate subset of the inputs and tying the remaining inputs either high or low. The Actel module can thus implement all two-input type logic functions, most three-input functions, and many four-input functions. A single module can also be used to implement a four input multiplexor and a latch register while two modules are needed to implement a single flip-flop.

The ACT<sup>TM</sup> 1 series FPGAs are available in two standard densities, the A1010A with 295 total logic modules and the A1020A with 547 logic modules. Both devices were chosen for implementation of the Omniview image processor core design. The Actel architecture consists of rows of logic modules separated by horizontal routing channels. The organization is similar to a channeled gate-array or standard cell approach except that instead of custom metallization areas between horizontal rows of modules, the Actel FPGA contains wiring segments of various lengths which can be connected through anti-fuses.

The Actel A1020A device can implement up to 547 latches and up to 247 flip-flops in a 44-pin, 68-pin, or 84-pin package. Both devices can provide a total of up to 57 input, output, or tri-state type interface pins.

### **Pythagoras Processor Discussion**

The U and V vector generators provide a two-dimensional raster-type scan to generate the equivalent of the object plane coordinates. The format of the U and V vector generator outputs is two's complement. The U and V outputs are passed through a two stage coordinate transformer which provides the square root of the sum of the squares of the inputs. The TMC2330, provided by TRW LSI Products Inc., converts bi-directionally between Cartesian (real and imaginary) and Polar (magnitude and phase) coordinates and can operate at up to 25 MHz.

The Omniview system uses the TMC2330 in the Rectangular-To-Polar transform mode. A functional block diagram of the TMC2330 is shown in Figure 4-5. The TMC2330 device is capable of other operations, including digital frequency synthesis using phase and frequency modulation. In the Rectangular-to-Polar (RTP) mode the device accepts two 16-bit two's complement or sign-magnitude integer coordinates in real and imaginary format. The transformation processor squares both inputs and generates the square root of the sum of the squares. The device also provides a 16-bit phase angle (modulo  $2\pi$ ) represent the arc tangent of the real and imaginary inputs. The internal pipeline of the TMC2330 requires 22 clock cycles between the input data and the output result.

In order to implement the equation for the RSS processor, two of the TMC2330 devices were used in the initial Omniview implementation. The first device takes the output from the U and V vector generators and generates the root-sum-square (RSS):

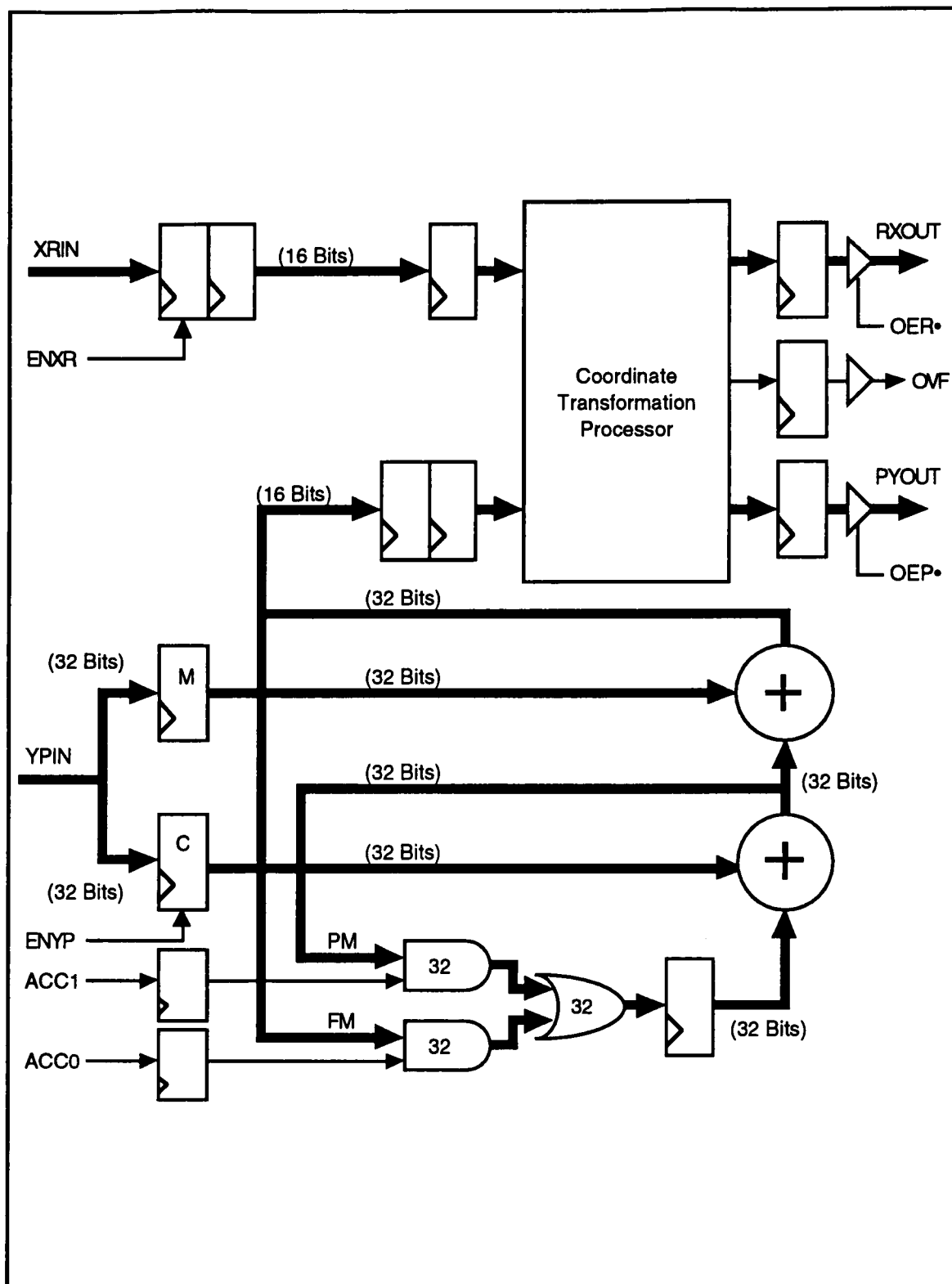


**Figure 4-5. Pythagoras Processor Functional Block Diagram**

$$RSS = \sqrt{U^2 + V^2} \quad (4.18)$$

The second device takes the RSS output of the first coordinate transformer and the output from the mR vector latch and generates the RSS to produce the final output:

$$RSS = \sqrt{(\sqrt{U^2 + V^2})^2 + (mR)^2} \quad (4.19)$$

The result in equation 4.19 can be simplified to the desired result as given by equation 4.17 above and is repeated here for convenience.

$$RSS = \sqrt{U^2 + V^2 + (mR)^2} \quad (4.20)$$

The denominator processor design allows the RSS processor to run separately from the numerator processor. The total pipeline delay from input to output of the RSS processor is 44 clock cycles. The pipeline control sequencer in the control logic block generates the sequence control for the numerator and denominator processor blocks so that the inputs to the high speed integer divider block agree and are in phase.

#### 4.4 HIGH SPEED DIVIDER IMPLEMENTATION

The final stage of the image processor design is accomplished by using a 32-bit high speed integer divider and an address offset and filter for each dimension of the transform. The use of a high speed integer divider allows the transforms to be implemented using the numerator and denominator approach as discussed in Chapter 3. The numerator term generated by each X and Y address processor is fed into the dividend input of each respective divider while the denominator term formed by the vector root-sum-square or RSS generator is input into the divisor input of both. The quotient output from the divider block is then passed through an address offset and filter block to further control the addressing of the video memory frame buffers.

##### High Speed Divider Logic

The TRW LSI Products' TMC3211 was used to provide the perspective scaling and perform the division operation. The TMC3211 is a fast monolithic two's complement fixed-point divider that can divide a 32-bit dividend by a 16-bit divisor to produce a full 32-bit quotient. A functional block diagram of the TMC3211 is shown in Figure 4-6.

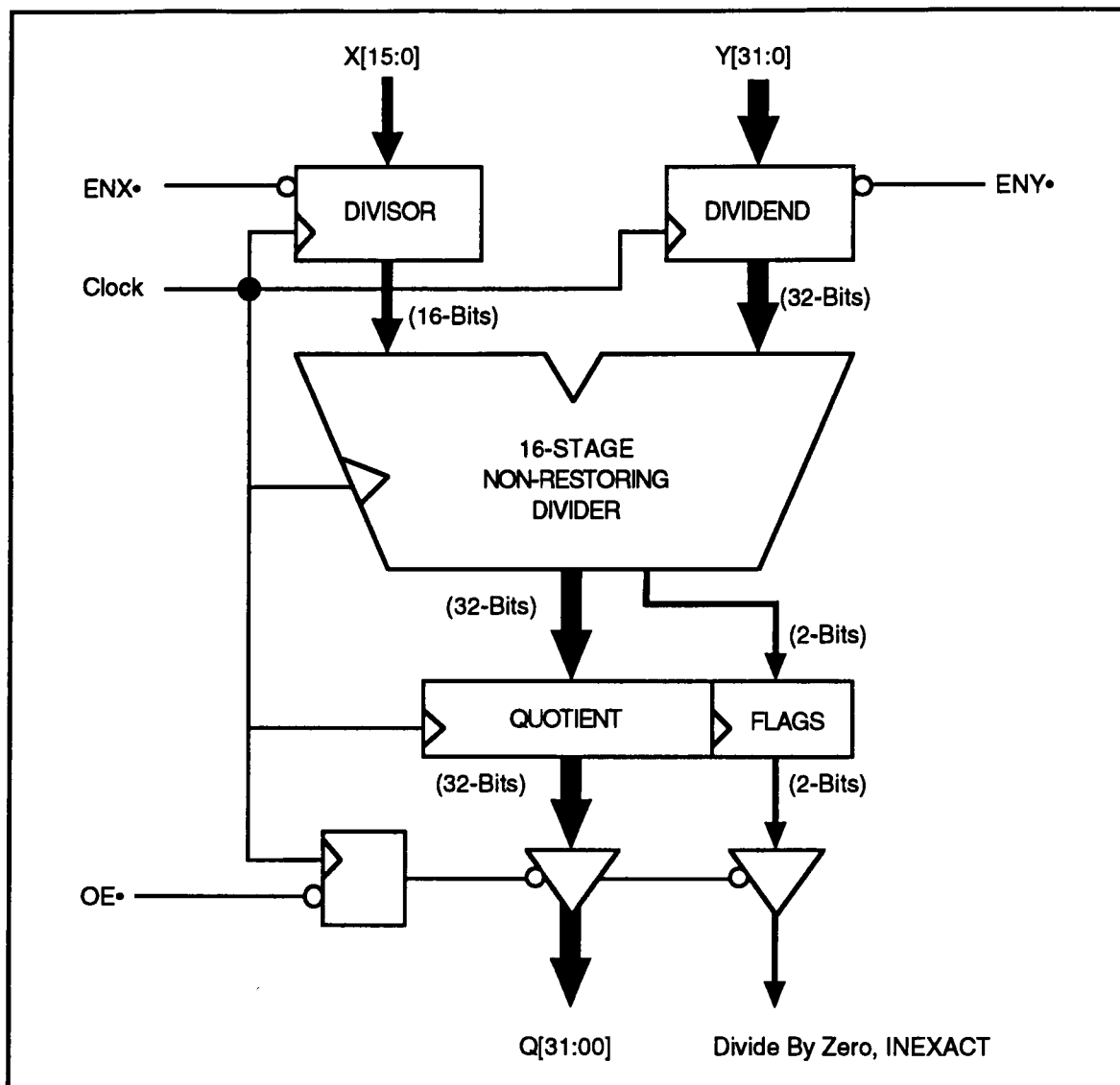


**Figure 4-6. 32-Bit Integer Divider Functional Block Diagram**

The TMC3211 uses an internal 16-stage non-restoring integer divider to provide a pipelined throughput of 20 Million Operations per Second (MOPS). The 32-bit dividend and 16-bit divisor data are input separately and synchronously with the rising edge of the clock input and have individual synchronous enable controls. The individual enable controls provide the ability to hold either input constant for a series of operations. The pipelined divider produces an output delayed exactly 19 clock cycles from the input data. The non-restoring divider array ensures that each division operation requires exactly the same number of clock cycles independent of the input data values. This is important in the

operation of the Omniview system since the architecture relies on constant pipelined throughput. The TMC3211 also provides two flags to indicate a divide by zero condition or an inexact result.

### **Address Offset Filter Generator**

After the scaling by the 32-bit by 16-bit integer divider, it is desirable to perform some address filtering operations on the X and Y address before they are used to address the input image buffer. There are two main reasons for implementing the address filtering. The input image from the fisheye or wide angle equipped imaging camera contains a circular image. The positioning of the image with respect to the exact center of the input image buffer as addressed by the image processor is critical. The image transform equations assume a two's complement addressed input buffer where the image is circularly symmetric and centered in the frame buffer. The address filter logic provides the ability to offset the image center as addressed by the image processor output in order to compensate for any offsets in lens-to-camera offsets and for shifts that can occur in the genlock and digitizing implementation. The other main function of the address filter logic is to provide some anit-aliasing address filtering. The address filtering is accomplished by rounding the address presented to the video memory buffers. The initial approach for the Omniview system provides only the image offset and image scaling functions. The filtering approaches were not implemented in the first pass of the design because it was not necessary for proof of the Omniview operation. The major concern was to be able to adjust the scaling and the image center to obtain the best transformation and nearest-neighbor sampling of the input image. Since the filtering logic is designed using a single high performance FPGA, the ability to perform other operations including filtering were supported.

The Address Offset and Filter block (AOF) was designed using a single 68-pin Actel FPGA. A functional block diagram of the AOF is shown in Figure 4-7. The AOF uses a 12-bit write-only host interface. The AOF contains two banks with two sets of selectable parameter registers. The internal parameters implemented in the first pass of the design are a shift value and an offset. The current set of parameters to use for the image transformation are selected by a View input. The View input is pipelined in order to compensate for the internal pipeline delay of the address path. The input to the AOF is a 26-bit address from the image processor and is input to a 26-bit pipeline register bank.

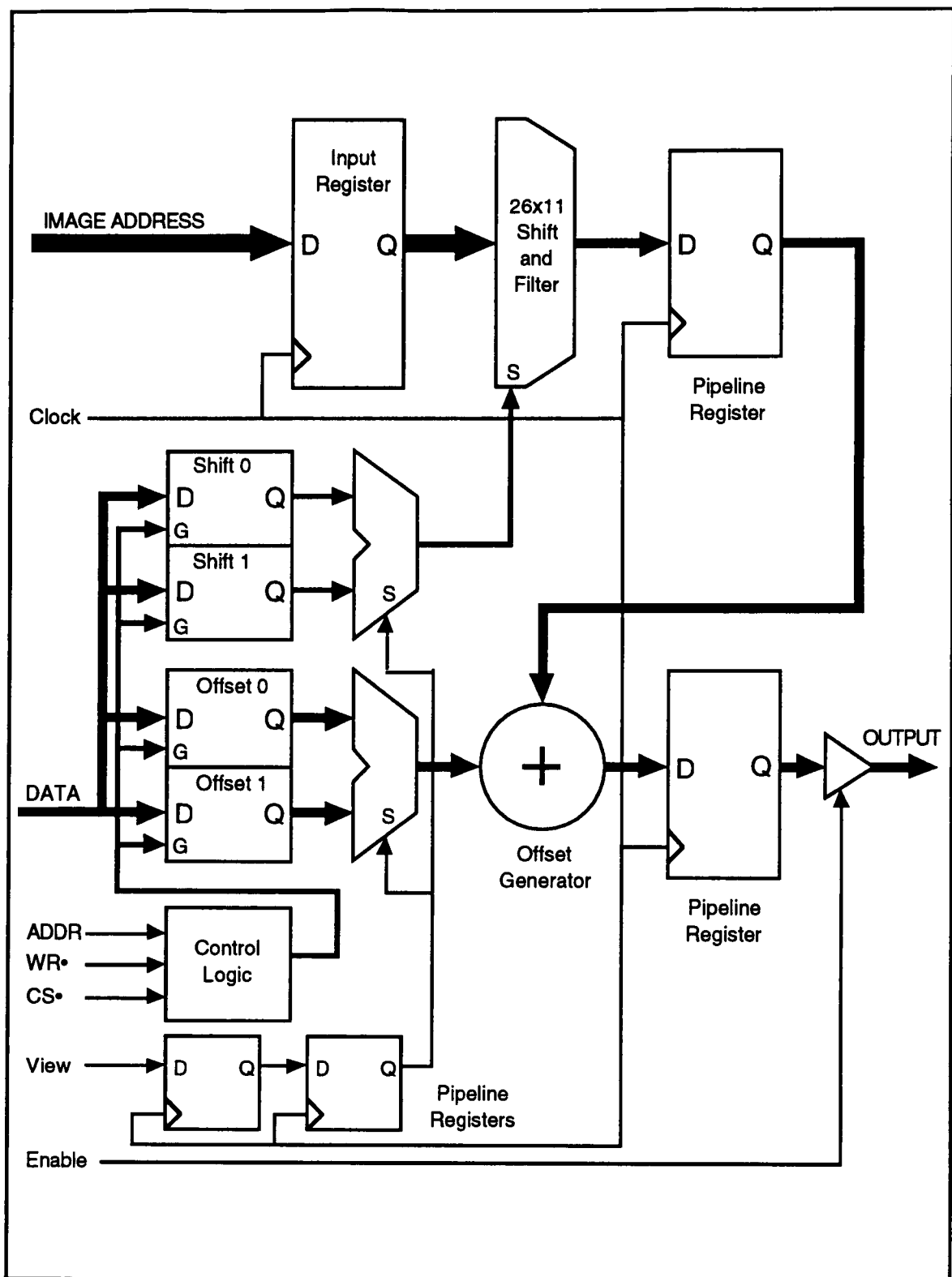


Figure 4-7. Address Control Functional Block Diagram

The output from the pipeline register passes through a 26-bit input to 12-bit output barrel shifter and rounding control logic to a second 12-bit pipeline register bank. The output from the second pipeline register bank is passed through a 12-bit adder to a third pipeline register bank. The address offset parameter is input to the 12-bit adder and added to the current shifted address to form the address offset. The pipeline register approach ensures maximum performance and minimizes the result of any logic delays. The pipeline approach provides the best performance for high performance arithmetic processing.

Using the current simulation results from the Actel FPGA design and simulation software, the expected throughput of the AOF is 20 MHz which is more than double the desired goal of 10 MHz to support standard 640 by 480 square pixel NTSC video transformation rates.

#### **4.5 CONTROL LOGIC DESIGN**

The image processor control logic provides the control and sequencing operation for the image processor board. The image processor board control logic consists of five main, functionally separate units - the address decode logic, the control register, the pipeline control sequencer, the clock generator and read/write control logic, and the vector clock and view parameter pipeline sequencer. A functional block diagram of the control logic core is shown in Figure 4-8.

##### **Address Decode and Control Register**

The address decode logic for the image processor board is implemented using a single 22V10 architecture style PLD. A PLD was used for convenience and allowed extra control logic functions to be included in the design. The image processor control logic consists of write only devices so the design was simplified by implementing a write only register type design. A memory map of the image processor board is shown in Figure 4-9. The device address decode logic decodes the host address lines to implement eight 1024-byte memory mapped address locations for each of the image processor functional control blocks. The main functional control blocks are: the control register, the Y address processor, the X address processor, the Y offset generator, the X offset generator, the V vector generator, the U vector generator, and the mR vector latch register. The control devices on the image processor board are additionally divided into two separate View parameter registers - View 0 and View 1. Each view contains identical control registers and

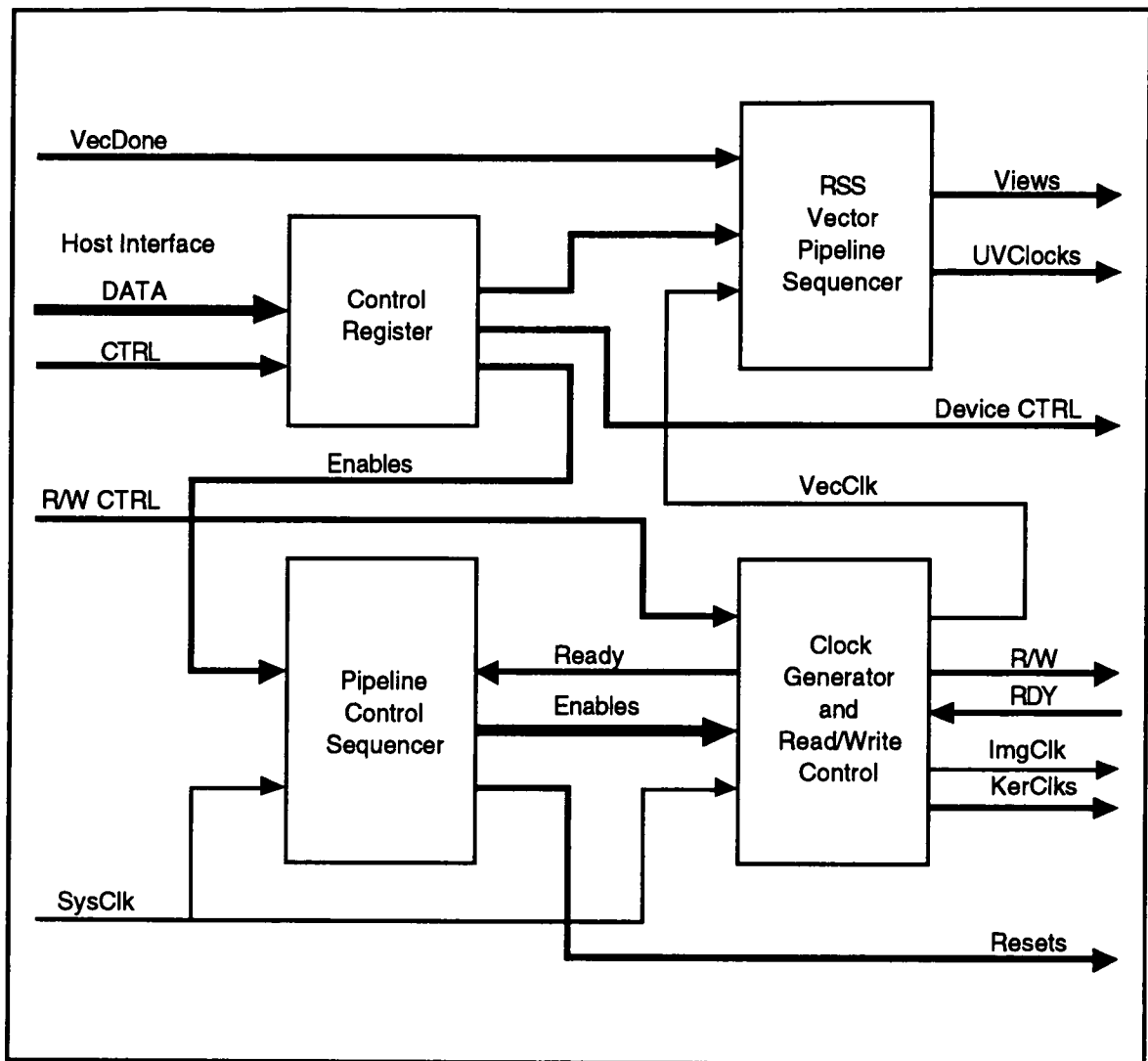


**Figure 4-8. Control Logic Block Diagram**

can be programmed independently. The use of dual View registers allows the image processor to support multiple transform windows during operation through pipelining. As one set of registers is being used, the other set of registers can be programmed by the host processor for a new transformation. A View select signal controlled by the pipeline sequencer logic selects the active window at the appropriate time during the transformation.

The control register consists of a 16-bit write-only register latch. The main function of the control register is to allow the host processor to start and stop the image processor in between image transforms. The control register also contains the View control bit which determines which set of parameters is active for the current transform.

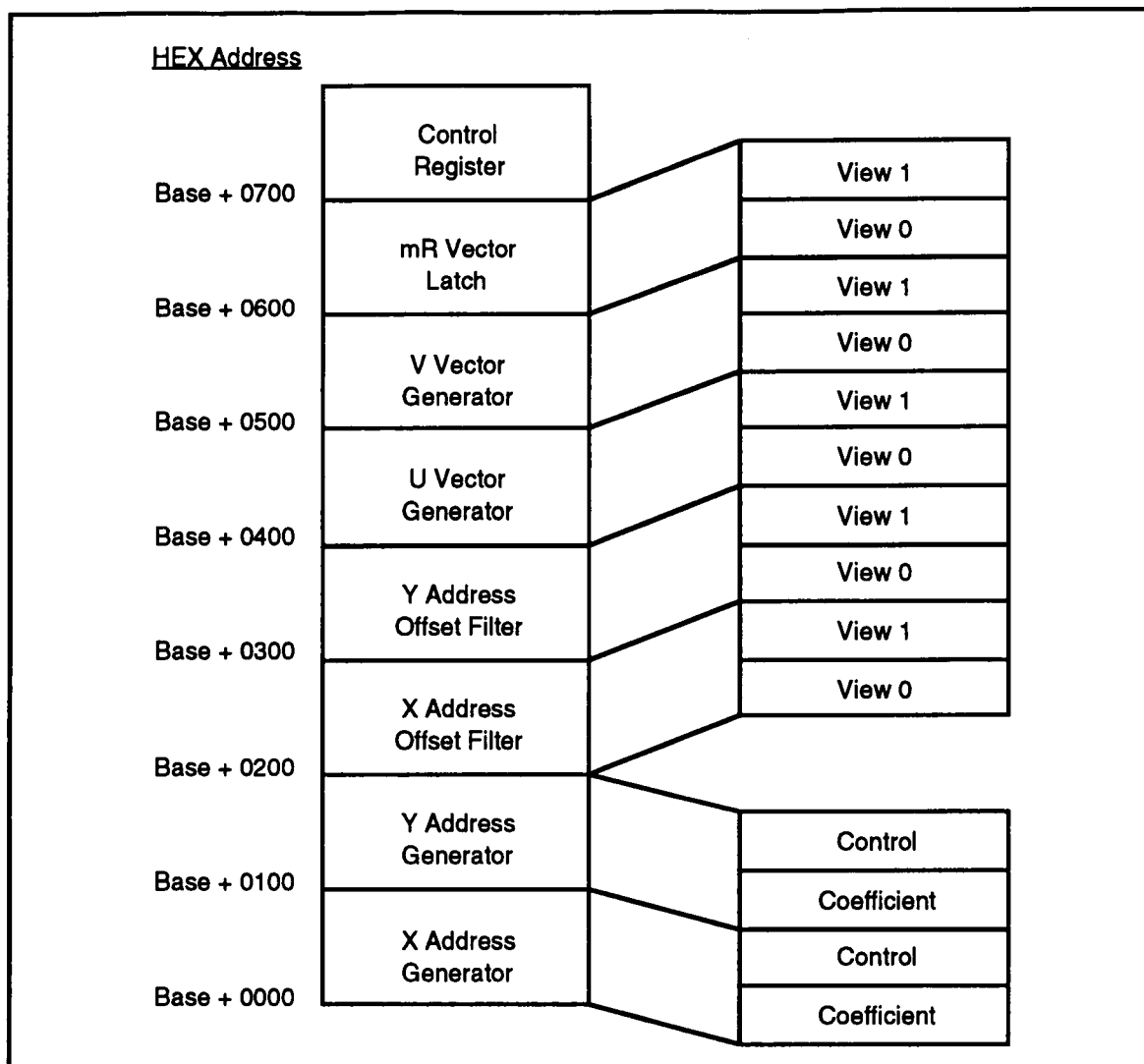


**Figure 4-9. Image Processor Memory Map**

### **Pipeline Control Sequencer**

The pipeline control and read/write logic are implemented using three special purpose programmable devices. The main sequencing of the image processor pipeline processing components is implemented using the Advanced Micro Devices Inc's Am29CPL154 field-programmable controller (FPC) device. The Am29CPL154 FPC is a CMOS single-chip field-programmable controller. A functional block diagram of the device is shown in Figure 4-10. The FPC core consists of four functional control units - the input block, the instruction decode logic, the program memory and pipeline register,

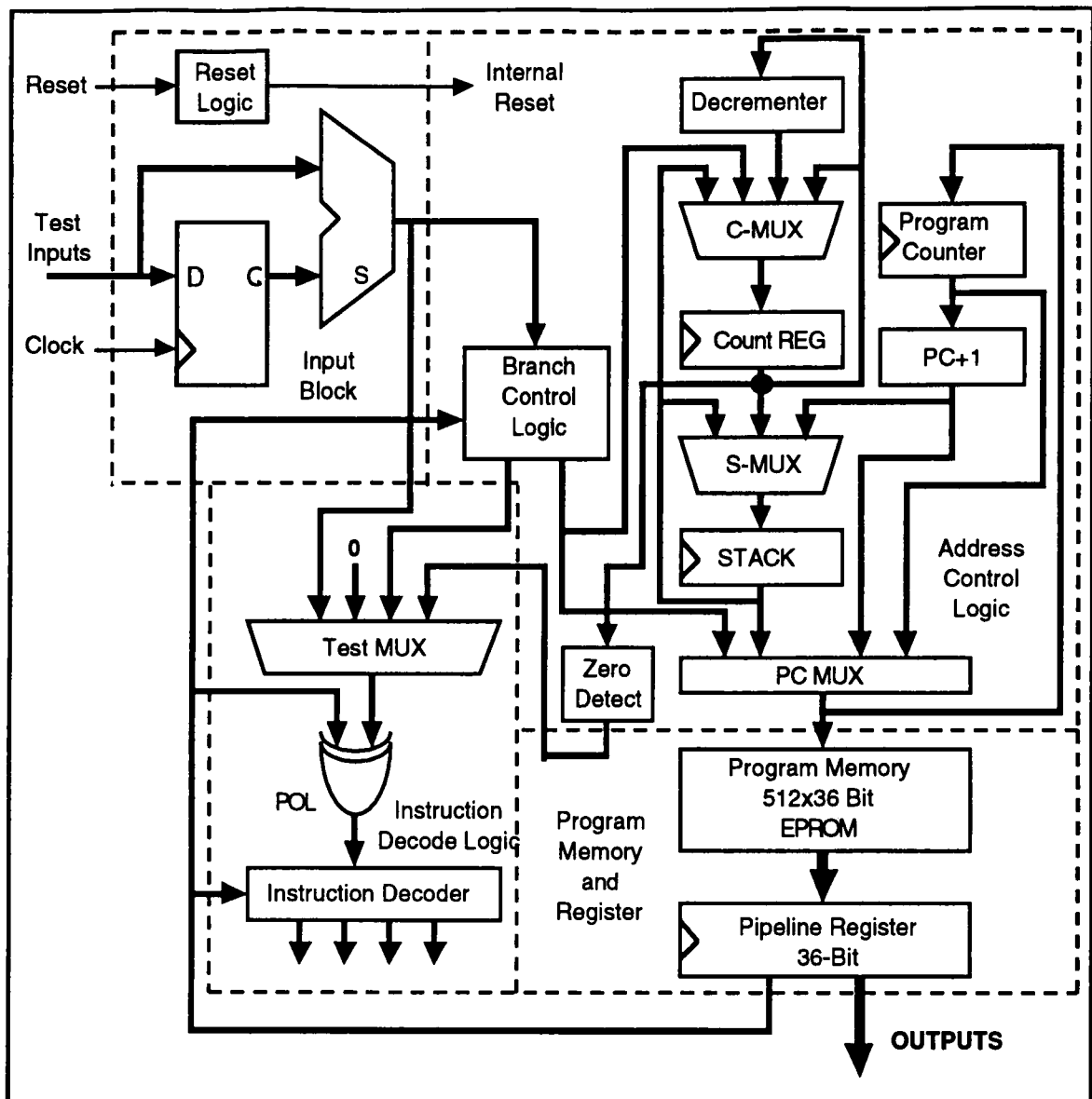


**Figure 4-10. Field Programmable Controller Functional Block Diagram**

and the address control logic. The FPC is implemented in CMOS technology and can operate at up to 33 MHz.

The input block provides an eight-input test block with individual registers and multiplexors for each input. Each of the conditional test inputs can be programmed as a registered synchronous or non-registered asynchronous input. Conditional test conditions can provide branching and loop control using the logical condition of a single or multiple test inputs.

The address sequencer is the heart of the FPC and provides the address sequencing to the internal 512-word by 36-bit program memory block. The address sequencer consists of four sub-blocks that provide flexible address generation and control sequencing. The four sub-blocks within the address sequencer are the stack, the program counter control, the counter block, and the branch control logic. The PC provides the same function as the instruction register in a general purpose microprocessor. The program counter block consists of a 4-to-1 multiplexor, a program counter (PC), and a 9-bit combinatorial incrementer (PC+1). The PC multiplexor selects the PC, PC+1, the branch address, or the stack as the next address to input to the program memory.

The stack control block consists of a 17-deep, 9-bit-wide stack with a 3-to-1 multiplexor (S MUX). The S MUX selects either the PC+1 output, the previous stack, or the output of the counter block to store in the next top-of-stack location. The stack is implemented as a push-pop type of stack. The stack register is automatically incremented after a value is written onto the stack (post-incremented) and decremented after a value is read from the top of the stack. The stack provides the ability to implement counter loops and to save branch and return addresses.

The counter sub-block of the address sequencer control block consists of a 9-bit, four-to-one multiplexor, a 9-bit counter register, zero detection logic, and a six-bit combinatorial decremter. The counter multiplexor (C MUX) selects between the top-of-stack, the branch control output, the decremter, and the counter register value as the next input to the counter register. The counter sub-block provides the ability to perform timing functions and iterative loop counting.

The branch control (GOTO) sub-block of the address sequencer provides three main functions to implement various branching functions. The GOTO block provides a 9-bit count value from the data field in the pipeline register or a masked input value to the counter block, a branch address from the data field to the PC MUX block, or a conditional input to the test multiplexor in the instruction decode block. The GOTO block allows masking of the conditional test inputs so that multiple inputs can be tested in a sum-of-products manner.

The program memory block contains the program for the operation of the FPC. The program memory consists of a 512 word by 36-bit EPROM with a 36-bit pipeline

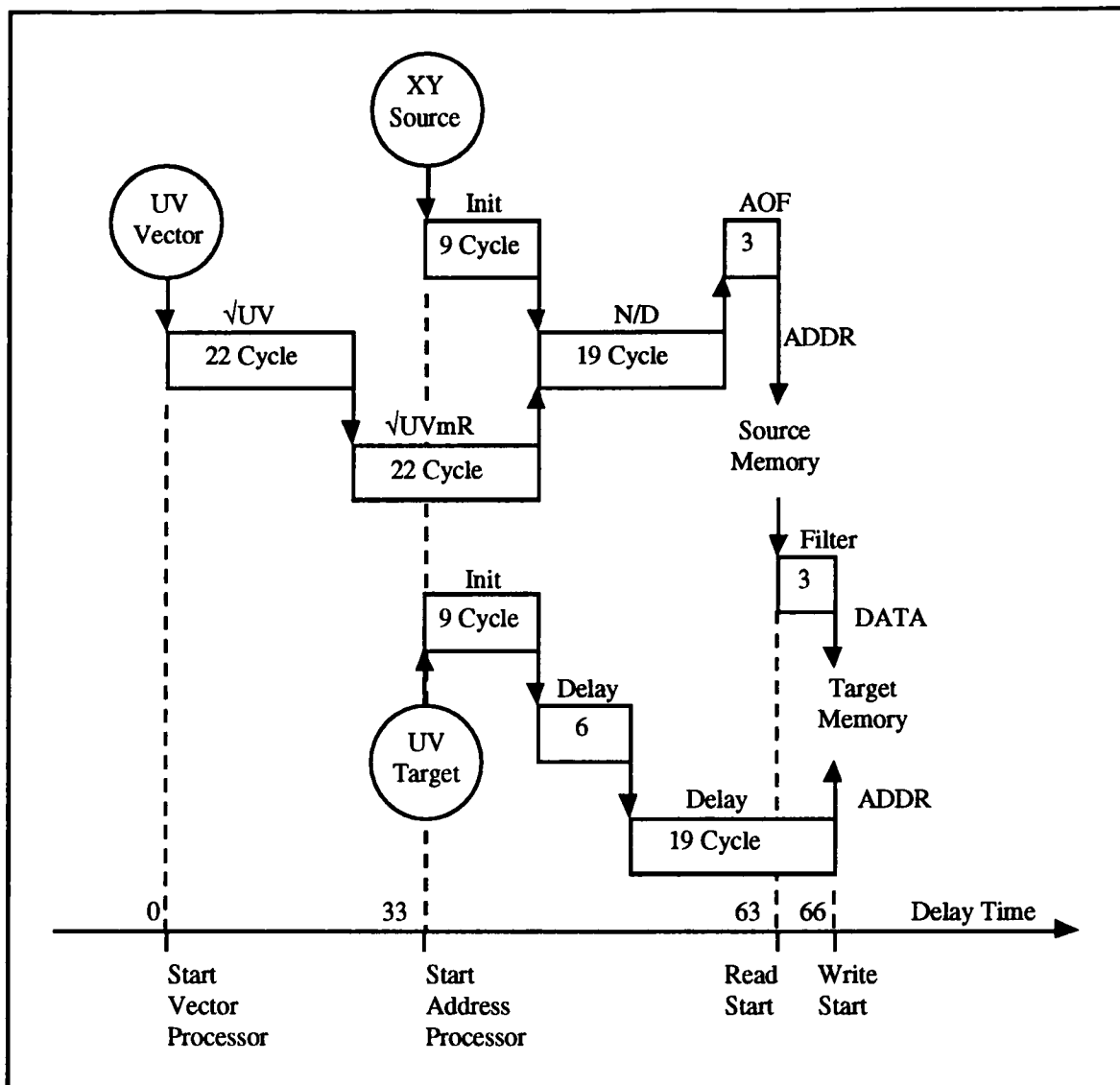
register. The program memory contains the user-defined instruction flow and output control sequence for the FPC. The upper 20 bits of the pipeline register are internal to the FPC and form the instruction word to control address sequencing. This functions as the instruction for the FPC. The 20-bit instructions contains several parts: an output enable bit, a five-bit opcode, a conditional test polarity select bit, a four-bit conditional test input select field, and a nine-bit immediate data field. The data field is used to provide branch addresses, counter values, and conditional test input masks.

The lower 16-bits of the pipeline register are brought out externally to provide user-defined, general purpose control outputs. The upper eight outputs can be tri-stated with the output enable bit of the control word while the lower eight bits are always enabled. The 16-bit output functional is defined as a part of the complete control word and is output whenever the control word is selected in the program flow. The FPC can execute all instructions including control branching in a single-clock cycle, therefore, the outputs can provide individual control sequencing functions each clock cycle.

A graphical representation of the pipeline initialization sequence is shown in Figure 4-11. The FPC pipeline sequencer controls the startup initialization and sequencing each of the image processor blocks. The vertical arrows and circles in Figure 4-11 represent data flow into or out of a pipeline processing element such as the high speed divider. The horizontal blocks represent a particular pipelined processing element. The timeline in Figure 4-11 shows the initial processing startup delay for the image processor. The figure is not necessarily drawn to scale and is meant to show the amount of pipelined and parallel processing implemented by the Omniview image processor.

### **Clock Generator and Read/Write Control Logic**

The clock generator control logic and read/write operation for the image processor is implemented using a single ICT Corporation PA7024 programmable logic device. The PA7024 is a CMOS 24-pin electrically erasable PLD with an extremely flexible internal architecture. The PA7024 provides 20 programmable input/output pins and two input/clock pins. Each pin programmed as an input can be either registered or latched. The internal register array contains 20 programmable logic function cells. Each internal logic cell can be programmed as a D, T, or JK type flip-flop with independent programmable



clock, reset, and preset product terms. The internal logic array provides up to 80 total sum-of-product functions.

The PA7024 was used to implement the main clock control generator and the read/write control logic for the image processor board. The system clock for the image processor board is provided by the host interface connector and operates at a 20 MHz frequency. The clock generator is driven by five separate signals from the pipeline sequencer block. Two enable clock signals from the pipeline sequencer control the sequencing of the function clocks. The read/write interface from the image processor is

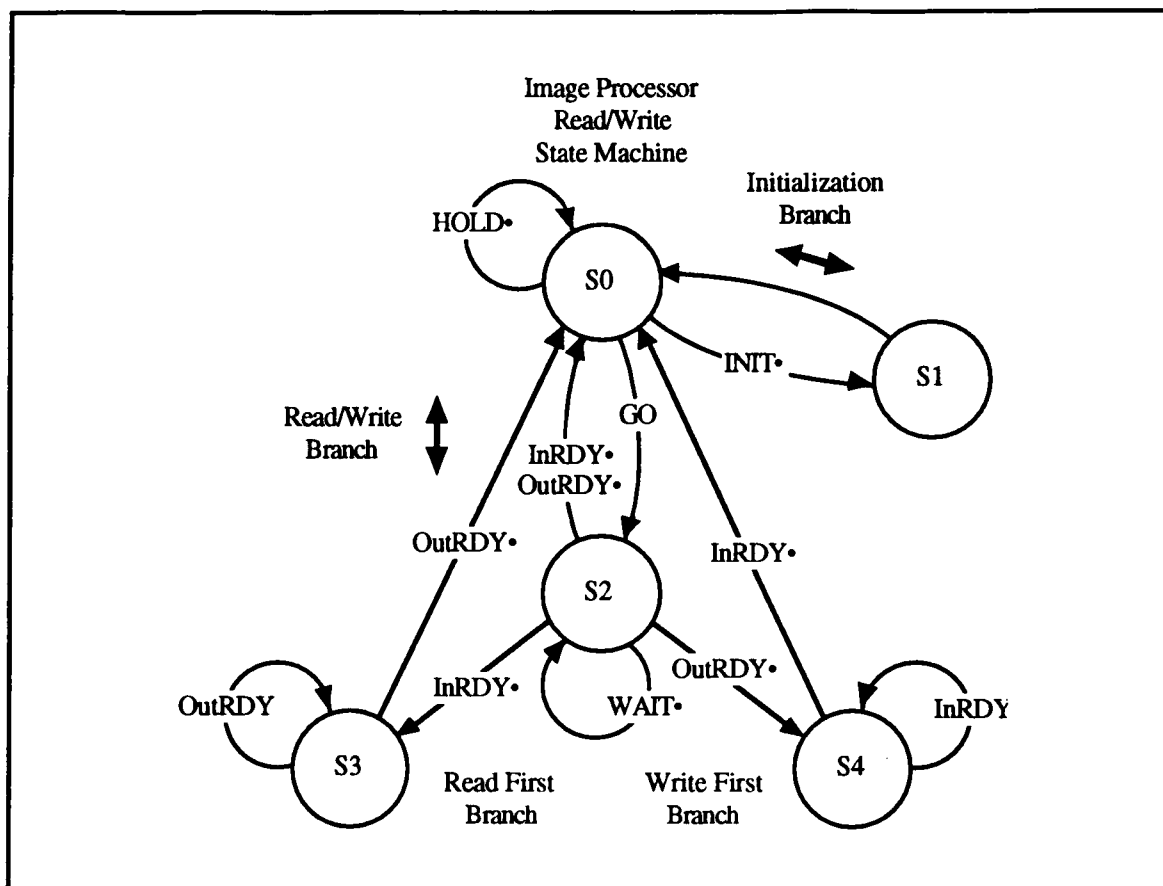


**Figure 4-12. CGC State Machine Diagram**

provided by two sets of handshake signals -  $\text{ImgREAD}\bullet$ ,  $\text{InRDY}\bullet$ , and  $\text{ImgWRITE}\bullet$ ,  $\text{OutRdy}\bullet$ . The CGC main control state machine diagram is shown in Figure 4-12. The CGC state machine uses two main branches. One branch is selected during startup to allow the image processor pipelined processing components to initialize. During the initialization branch, video memory accesses are disabled. The second branch provides the read and write access arbitration. When the system is first powered on, the first branch is used to allow the image processor core to fill up all of the pipelined processing and delay elements.

The CGC provides all of the function clocks to the rest of the image processor system. The  $\text{ImgClk}$  signal provides the XY address processor clocks, the  $\text{VecClk}$  provides the vector generator clock, and the  $\text{KerClk}$  provides a FIFO clock to support pipeline delay for the filter coefficient address. Additionally, three clocks ( $\text{IntClkP}$ ,  $\text{IntClkXY}$ , and  $\text{IntACC}$ ) provide sequencing of the interpolator filter logic that resides on the video processor board. The  $\text{IntClkXY}$  is used to clock the input buffer data and

coefficient into the filter logic while the IntClkP signal is used to clock the accumulated products for the filter core. The interpolator filter products are accumulated as long as the IntACC signal is high.

### **Vector Clock Generator and View Pipeline Control**

The Vector Clock Generator (VGC) provides the clock control for the UV-vector generator block. The input clock is the VecClk signal generated by the CGC. The VGC also provides the pipeline delay timing for the view parameter register select. Four individually delayed view signals are used in order to time the sequencing of the view parameter registers: TarView, SrcView, mRView, and VecView. The SrcView is used to select the parameter register set for the address offset and filter logic. The VecView signal selects the current vector parameter register set for the UV-vector generator block. The mRView selects the current magnitude vector parameter register. Finally, the TarView signal is used to select the convolution filter logic coefficient parameter register set. All of the View control signals enable the image processor to operate continuously while alternating between image window parameters. This allows the implementation of multiple image windows to be displayed on the output display monitor with each window providing different viewing parameters such as pan, tilt, rotation, and magnification.

## **4.6 SUMMARY**

The design and implementation of the image processor core was discussed. The main emphasis in this chapter is the discussion of the key elements of the image processor design. Several unique hardware components were used in the implementation. The advanced address generator or AIMS provides the capability to generate 3rd order two-dimensional transformations and operates at frequencies up to 40 MHz. The AIMS provides the numerator processor for the Omniview transformation implementation.

The denominator section of the image processor was implemented using some special FPGA designs and a unique pythagoras type arithmetic processing element. The TMC2330 provides the capability to generate the square root of the sum of the squares of two 16-bit two's complement inputs. The Omniview design uses two of the TMC2330 pythagoras processor in order to generate the final vector function of the Omniview transformation algorithms.

A very fast 32-bit by 16-bit integer divider was used to generate the X and Y addresses from the numerator and denominator section outputs. The divider was accomplished using the TMC3211 which can operate at a frequency up to 20 MHz.

The final X and Y image addresses were formed by passing the output of the high speed integer divider through an address filter and offset generator (AOF). The main purpose of the AOF is to allow adjustment for the image center within the input image memory buffer and to shift the address range to compensate for binary point scaling. The AOF was implemented using Actel field-programmable gate arrays.

The image processor was implemented using a single printed circuit board and operates at a system clock frequency of 20 MHz. The processing bandwidth of the image processor is sufficient to support standard NTSC 640 by 480 square pixel display images.

## **CHAPTER 5**

### **OMNIVIEW HOST PROCESSOR DESIGN**

The host processor environment for an embedded design such as the Omniview system must handle many tasks. Several considerations are necessary in order to insure adequate real-time response during operation. The Omniview host microprocessor provides the interface between operator control inputs and the image manipulation and graphics functions as well as communication with external control devices.

A goal of the Omniview host microprocessor design is to provide the capability to perform real-time image stabilization for multiple images at real-time rates. Image stabilization is necessary for applications such as the Unmanned Aerial Vehicle (UAV) remotely piloted airplanes and helicopters used by the military and law enforcement agencies for remote airborne reconnaissance. In these types of applications, the Omniview system would provide the capability to stabilize the video images to compensate for wind effects and trajectory paths in order to maintain a fixed position for target viewing. For instance, during a mission the pilot would use a primary image for navigation and flying the UAV while another operator might use a secondary image for target acquisition and surveillance. Each image might be stabilized independently for optimum viewing situations.

In order to support these different real-time image manipulation scenarios, the Omniview host microprocessor must provide enough processing power to perform real-time calculation of the image processing transformation parameters for multiple images. Another goal for the Omniview camera system was to provide the ability to display and manipulate up to four images at any one time. Therefore, the host microprocessor should be able to update these four images up to 30 times each second for a standard 30 frame-per-second video camera system giving an update throughput for the host of 120 image parameter updates-per-second. Another consideration for the host microprocessor was the

desire to implement all of the image transformation coefficient calculations and pan/tilt/zoom viewing angles using hardware floating-point arithmetic.

A functional block diagram of the final Omniview host microprocessor core design is shown in Figure 5-1. The Omniview host core contains many of the standard components necessary for a stand-alone embedded microprocessor system. Other considerations affecting the design and choice of components are cost, reliability, and availability. Several features of the Omniview host microprocessor core that are unique to embedded applications are discussed here.

## **5.1 HOST MICROPROCESSOR ARCHITECTURE**

The Intel 80960SB microprocessor was chosen as the host microprocessor for the Omniview camera system implementation because it meets the requirements for processing power, cost, and performance necessary to support the real-time image updates. The 80960SB architecture can be characterized as a high performance processing engine. A block diagram of the 80960SB microprocessor internal hardware architecture is shown in Figure 5-2. The 80960SB architecture provides many design enhancements over standard microprocessors to enhance the computational and data processing speed. The 80960SB microprocessor supports high-speed pipelined execution, register scoreboarding, burst bus read/write transfers, and extensive fault handling capabilities for debugging and monitoring. It is a RISC type 32-bit microprocessor with a 32-bit instruction set and set of local and global 32-bit registers. Another important feature of the 80960SB system is that it implements the IEEE-796 specification for floating point operations in hardware.

The 80960SB microprocessor internal architecture is composed of six main functional units as shown in Figure 5-2:

1. Bus Control Logic (BCL)
2. Instruction Fetch Unit and Instruction Cache (IFU)
3. Instruction Decoder (ID)
4. Micro-Instruction Sequencer and ROM (MIS)
5. Instruction Execution Unit (IEU)
6. Floating Point Unit (FPU)

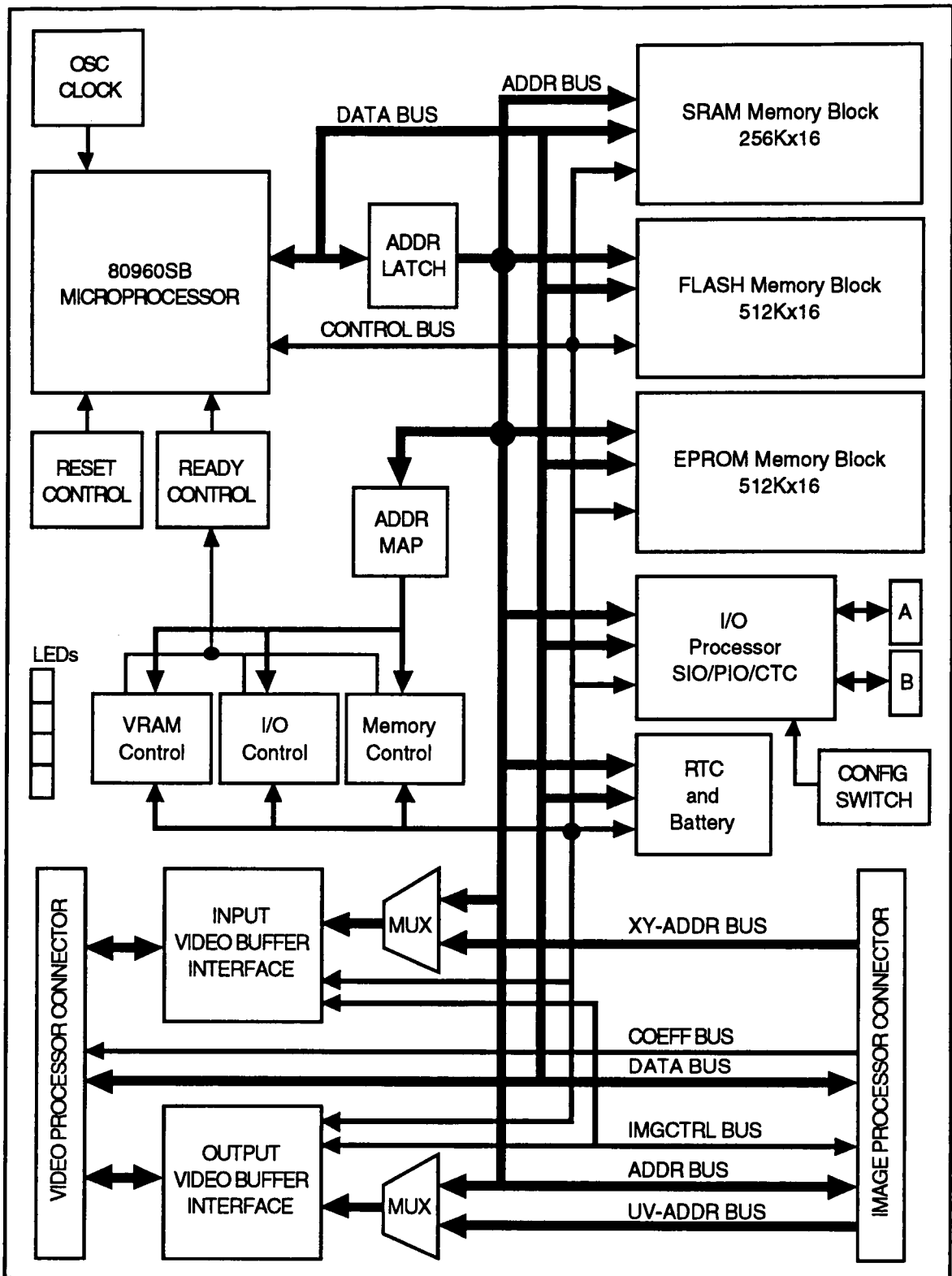


**Figure 5-1. Host Processor Functional Block Diagram**

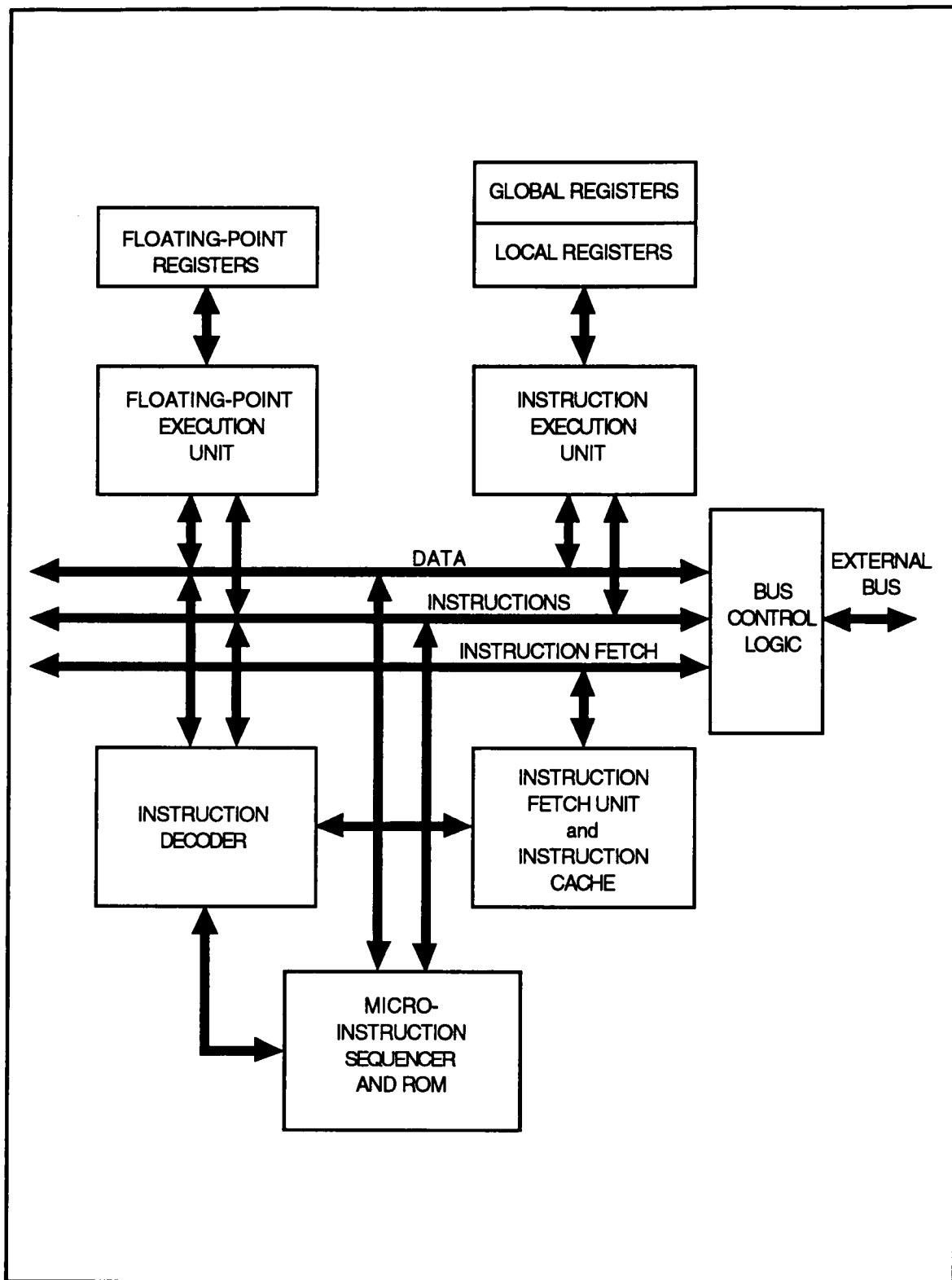


**Figure 5-2. Intel 80960SB Microprocessor Internal Architecture**

The Bus Control Logic (BCL) provides the interface between the processor and the external world. The BCL unit accepts requests from other units within the microprocessor and executes them while attempting to maximize bus access efficiency through buffering and burst accesses. The BCL unit implements a queuing mechanism that can buffer up to three outstanding requests at any given time. This queuing approach in coordination with the register scoreboarding of the 80960 allows other units within the microprocessor to continue operation without waiting for bus requests to finish. The BCL also generates burst bus cycles on the external bus allowing up to 16 bytes of contiguous data to be read or written in a single bus operation.

The Instruction Fetch Unit (IFU) along with its internal instruction cache provides intelligent buffering for the Instruction Decoder (ID). The IFU uses several mechanisms in order to present the instruction stream to the ID in the fastest and most transparent manner. A 512 byte direct-mapped instruction cache is maintained by the IFU allowing fast access to instructions. While other units in the 80960 are executing instructions, the IFU looks ahead in the instruction sequence stored in the instruction cache. If the IFU detects an instruction that is needed but not in the cache, i.e. a cache miss, it issues a prefetch request to the BCL.

The Instruction Decoder (ID) decodes the instructions as it receives them from the IFU and routes them to appropriate execution units. The ID decodes instructions into five main groups according to the way instructions are executed:

1. Simple Instructions
2. Floating Point Instructions
3. Branch and Fault-If Instructions
4. Complex Instructions
5. Load and Store Instructions

Simple instructions are decoded by the ID and passed to the Instruction Execution Unit (IEU). These instructions include logical, comparison, shift, and add/subtract and are usually executed in a single clock period. All floating-point instructions are passed to the Floating-Point Unit (FPU). Since many of the floating-point instructions are executed using microcode, the FPU must interact with the ID and MIS. The ID executes all branch instructions directly. Complex instructions are passed by the ID to the MIS where they are

executed by microcode. Load and store instructions are simple instructions used to read or write data to and from memory. The ID sends these instructions to the BCL which executes them.

The Micro-Instruction Sequencer Unit (MIS) and associated ROM is designed to help in the execution of instructions that must use microcode. All of the processor's microcode is stored in the ROM. Some examples of microcoded instructions are the floating-point trigonometric functions sin, cos, and tan. Other useful microcode routines that are stored in the ROM include the power-up and self-test sequence for processor initialization, and the interrupt and fault handling routines.

The Instruction Execution Unit (IEU) contains the Arithmetic Logic Unit (ALU) and the mechanism for register scoreboarding. It also manages the global and local register sets. The ALU performs all of its operations in a single clock cycle. The Floating Point Unit (FPU) executes all of the floating-point computations as well as the integer multiple and divide operations. It also manages the four 80-bit floating point registers used for extended precision floating-point calculations. The FPU can also use the global and local registers for floating-point operations. Finally, the FPU gets microcode from the MIS and ROM for execution of complex floating-point operations.

## **5.2 OVERVIEW OF EXECUTION ENVIRONMENT**

The execution environment for the 80960SB microprocessor includes many of the standard features of most microprocessor system. What makes the 80960SB unique is its register scoreboarding and over-lapped instruction execution.

The execution environment is shown in Figure 5-3. When the 80960SB is initialized via the external hardware reset it sets up an execution environment. Part of the execution environment for the 80960SB is shown in Figure 5-3. This environment consist of a  $2^{32}$  - byte address space, a set of global general purpose registers and global floating point registers, a set of local registers, a set of arithmetic control bits, a set of process control bits, a set of trace control bits, and the instruction pointer register. All of these items except for the address space are contained on the 80960SB device. The other main items of the execution environment are the system stack, the procedure stack, and the interrupt table.

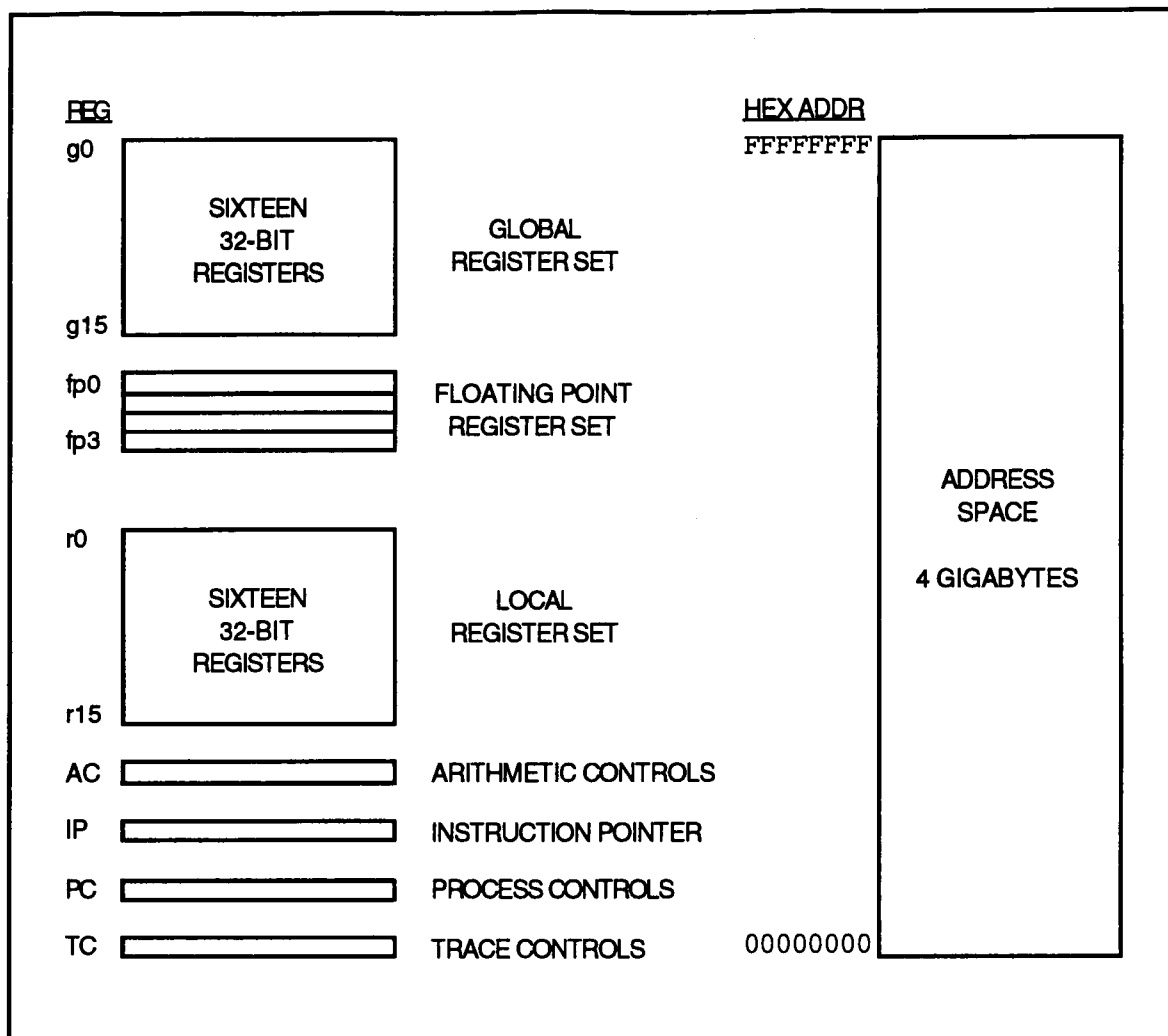


**Figure 5-3. Intel 80960SB Execution Environment**

There are three types of registers supported by the 80960SB architecture: global, local, and floating point registers. The 16 global registers (g0 through g15) constitute a set of general purpose 32-bit registers that are preserved across procedure boundaries. The local registers (r0 through r15) provide an additional set of sixteen 32-bit registers. The purpose of the local register set is to provide a separate set of registers, in addition to the global and floating point registers, for each active procedure. Each time a procedure is called, either by a system call or an interrupt, a new set of local registers is automatically allocated for the called procedure while the previous set is saved. This provides enhanced performance for procedures such as interrupts, traps, and subroutines because scratch registers are immediately available.

## Register Scoreboarding Operation

The 80960SB provides an additional speed enhancing mechanism called register scoreboarding that in certain circumstances can permit instructions to be executed concurrently. The register scoreboarding provides essentially parallel operation for some sequences of instructions. The mechanism for register scoreboarding uses scoreboard bits to indicate when a particular register or set of registers is being used by an instruction. If the instructions that follow do not use the same register or registers in the register group, the 80960SB microprocessor can in some instances execute those instructions before the execution of the prior instruction is finished. In effect, the register scoreboarding mechanism allows some instructions to be executed in parallel.

The register scoreboarding feature is illustrated by the following example. A register load operation typically takes 4 to 10 clock cycles or more (depending on the design of the system memory and the addressing mode used). Register scoreboarding allows other instructions to be executed concurrently with the load instruction if the subsequent instructions do not use the same set of registers as the load instruction. For example, the following group of instructions loads a group of local registers while performing some other operations on data in the global registers.

|      |                |                                |
|------|----------------|--------------------------------|
| LD   | addr1,r6       | # r6 ← data from address addr1 |
| ADDI | g4, g6, g7     | # g7 ← g4 + g6                 |
| ADDI | g9, g10, g11   | # g11 ← g9 + g10               |
| LD   | addr2, r8      | # r8 ← data from address addr2 |
| AND  | g0, 0xFFFF, g1 | # g1 ← g0 & 0xFFFF             |
| ADDI | r6, r8, r7     | # r7 ← r6 + r8                 |

For this example, the two ADDI (add integers) instructions following the first LD instruction and also the AND instruction following the second LD instruction are performed concurrently with the bus accesses of the two load instructions. In this example, therefore, the 80960SB microprocessor executes five instructions in the same time that a standard architecture microprocessor would take to perform the two load instructions alone. The register scoreboarding approach makes the Intel 80960 architecture very efficient.

## Overview of Floating Point Support

The 80960SB microprocessor provides a complete implementation of the IEEE standard for binary floating point arithmetic (IEEE 754-1985). This implementation includes a full set of floating point operations that include add, subtract, multiply, divide, trigonometric and logarithmic operations. All floating point operations can be performed on single precision (32-bit), double precision (64-bit), and extended precision (80-bit) real numbers and are performed by the floating-point unit (FPU). The FPU performs all floating-point computations for the processor as well as the integer multiply and divide operations. It also manages the four global 80-bit floating point registers which it uses for extended-precision, floating-point calculations. The FPU can share processor resources using global and local registers for operands in floating-point operations.

The FPU obtains microcode for the execution of complex floating-point instructions from the micro-instruction sequencer (MIS). One key benefit of the 80960SB is that the floating point processing is integrated into the normal execution environment. All floating point operations can reference as source or destination data contained in the global or local registers. Even though the general purpose and local registers are only 32-bit in length, operations on real numbers convert to 80-bit extended precision automatically and back to 32-bit or 64-bit results when the a general purpose register is used. Single-precision and double-precision real numbers are stored in the same general registers as ordinal numbers. In addition, the four global 80-bit floating point registers can be used to hold extended precision values.

### 5.3 OVERVIEW OF 80960 BUS OPERATION

The 80960SB is one version of the Intel i960™ family of RISC microprocessors and uses a 16-bit multiplexed address/data bus to connect to memory and peripherals within the 4 gigabyte address space. The main feature of the 80960SB bus that differs from other embedded microprocessors is its burst-transfer capability. The 80960SB provides the ability to transfer up to 32-bytes of data in 8 successive 16-bit word transfers at a maximum rate of one word every processor clock cycle.

A block diagram of the logical interfaces to the 80960SB microprocessor is shown in Figure 5-4. The address bus is 32-bits with the lower 16-bits of the bus multiplexed between address and data transfers. An additional three bits of the address bus A[3:1] are

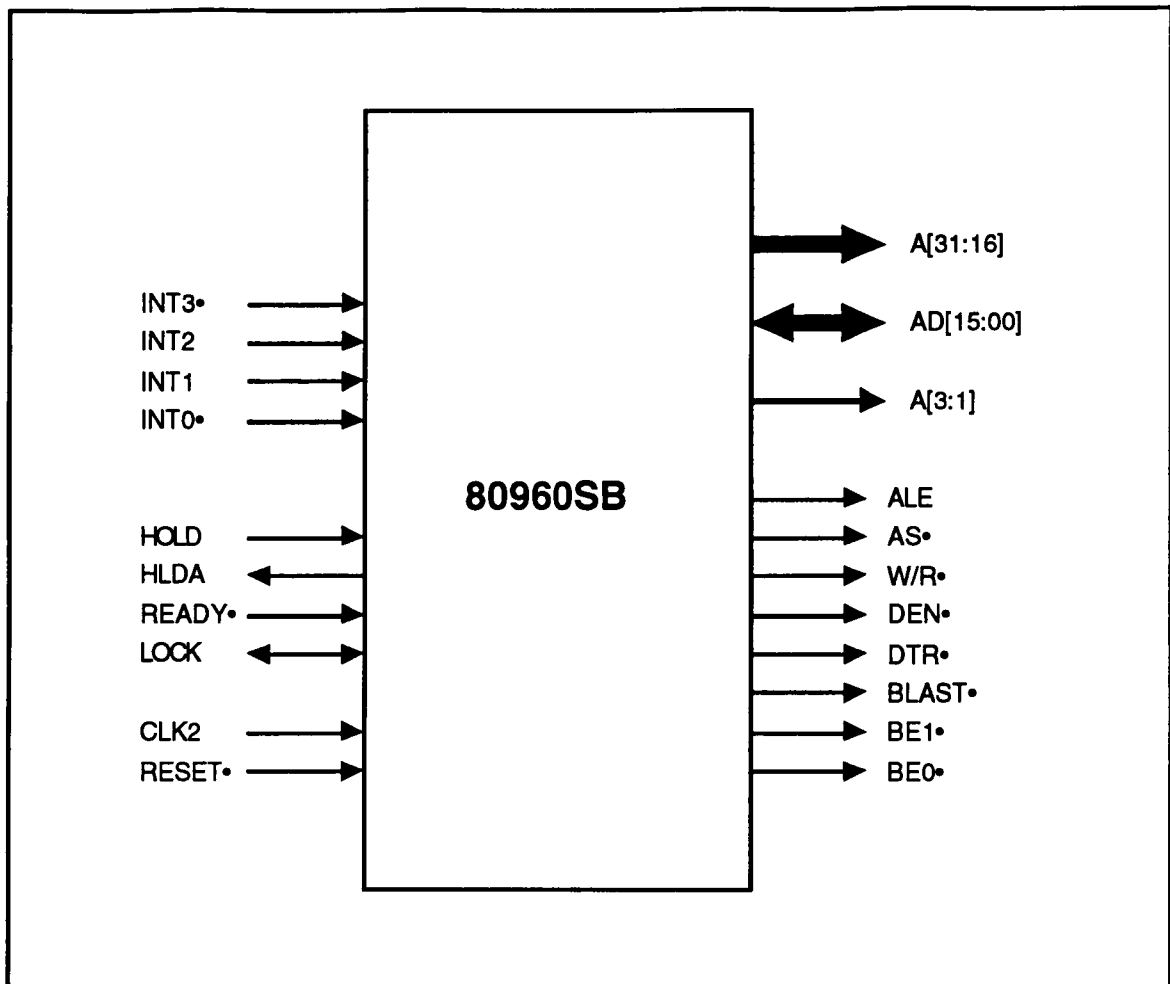


**Figure 5-4. Intel 80960SB Logic Interface Diagram**

also brought out externally to facilitate burst bus transfers. There are nine bus control signals that provide the control logic for interfacing to memory and peripherals. The operation of the control signals is explained briefly in the following paragraphs. The 80960SB uses four interrupt inputs for hardware interfacing. The INT0• and INT1 are dedicated interrupts and provide direct vectoring to interrupt routines. The INT2 and INT3• inputs can be used in either the dedicated vector interrupt mode, or they can be programmed to operate in an interrupt vector handshake mode. Two additional pins provide a hardware hold and acknowledge interface so that multiple bus masters may share the 80960SB address space. A brief description of the 80690SB interface bus operation is included in the next section to demonstrate the high-bandwidth operation of the processor and its ability to support real-time embedded applications.

A bus state diagram for the 80960SB bus operation is shown in Figure 5-5. There are five basic bus states the 80960SB can be in at any one time. These bus states are the idle ( $T_i$ ), address ( $T_a$ ), data ( $T_d$ ), recovery ( $T_r$ ), and wait ( $T_w$ ) states. During system operation, the processor continuously enters and exits these five different bus states.

The bus occupies the idle ( $T_i$ ) state when no address/data transfers are in progress. When the bus control unit receives a transfer request from the execution unit, the bus enters the ( $T_a$ ) state to transmit the address onto the bus. Following the ( $T_a$ ) state, the bus enters the ( $T_d$ ) state to either transmit or receive data on the multiplexed bus. If the system is not ready to complete the data transfer, the bus can be made to enter the wait state ( $T_w$ ) for as many clock cycles as necessary. The ( $T_w$ ) state may be maintained for as many clock cycles as necessary to allow sufficient time for memory or I/O devices to respond. When the data transfer is completed, the bus exits the  $T_d$  or  $T_w$  state and enters the recovery ( $T_r$ ) state in order to allow external memory or I/O devices time to recover from the transfer operation. The bus then reenters the  $T_i$  state if no new bus request is pending. If a new request is pending when the bus exits the  $T_r$  state, the bus reenters the  $T_a$  state to transmit the new address and start a new bus cycle.

### **Overview of Burst Transactions**

One of the key features of the 80960SB architecture is its burst-bus operation. The 80960SB microprocessor supports burst-bus transactions that can read or write up to 16 contiguous bytes or eight contiguous 16-bit words of data at a maximum rate of one 16-bit word every bus cycle. For the Omniview system with a 20 MHz microprocessor clock frequency, the bus can transfer data at 10 MHz rates. Since the can bus can transfer a full 16-bit word during each transfer cycle, this corresponds to a bus rate of 20 million bytes per second. This burst-bus feature coupled with the 80960SB internal cache architecture enhances the 80960SB microprocessor's performance significantly.

The bus control signals for the 80960SB are designed to provide easy integration into standard memory architectures. A burst read or write transaction is very similar to a standard read or write transaction with a few differences in control signal handshaking. The main difference between a burst bus transaction and a single-word transaction is that the burst transaction transfers up to 16 bytes per access cycle.

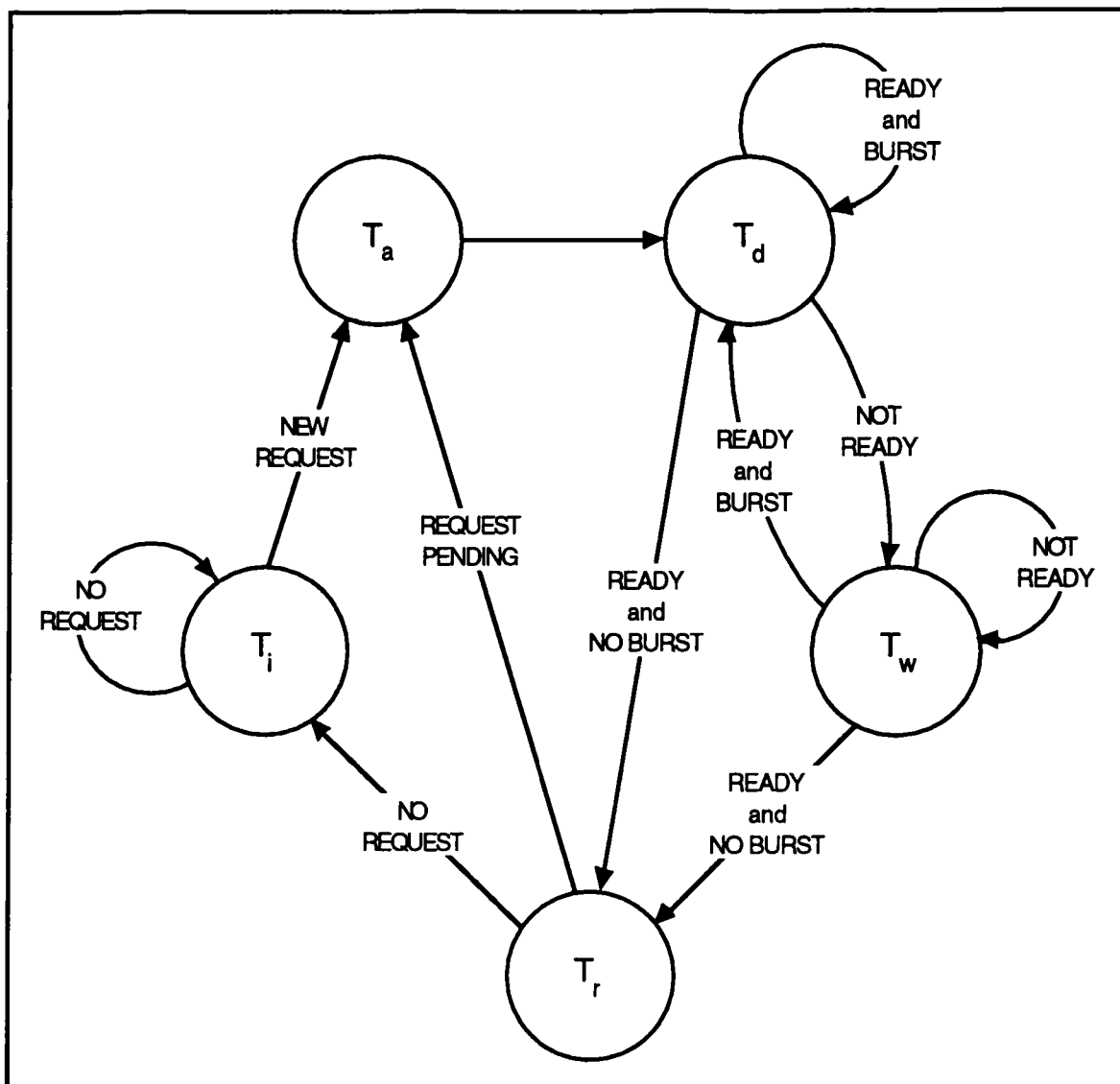


**Figure 5-5. Intel 80960SB Bus State Diagram**

An example of a zero wait state 16-byte burst-read cycle is shown in Figure 5-6. The 80960SB enters the address state  $T_a$  during the first cycle of the transaction as in any read or write bus cycle. The 80960SB signals the address state by bringing  $AS\bullet$  low for the first cycle. The address is extracted from the multiplexed address/data bus and latched externally with the address latch enable pulse  $ALE$ . Three signals are used to ensure that bus transceivers and peripheral I/O devices tied to the bus do not drive during the address state. The transaction is defined as a write or read operation by the  $W/R\bullet$  signal. Also, the data transfer/read ( $DT/R\bullet$ ) signal indicates the direction of the transfer for the transaction

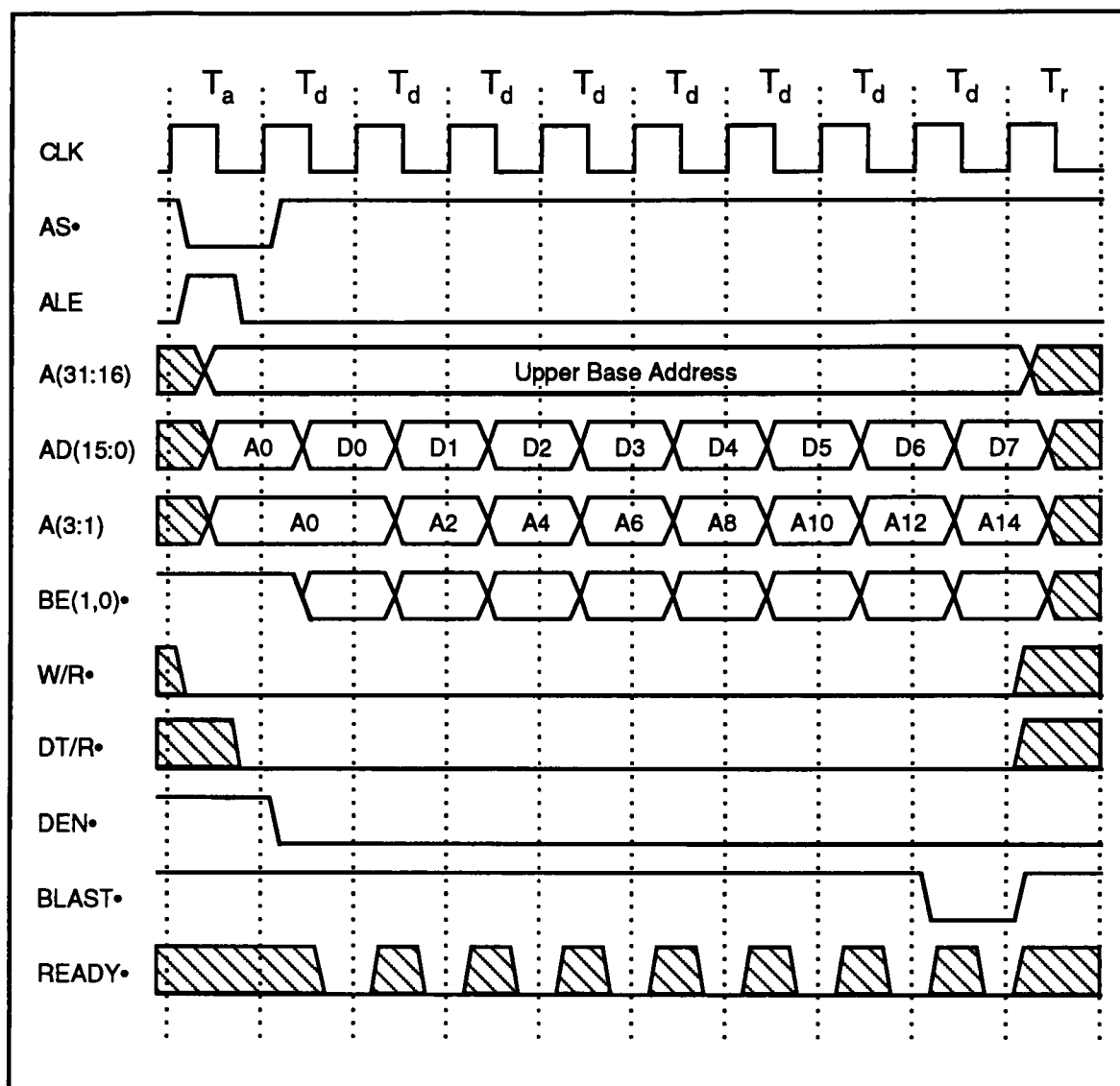


**Figure 5-6. Burst Transaction Timing Diagram**

operation and corresponds logically to the W/R\* signal. The DEN\* signal is asserted when the address state ( $T_a$ ) is exited and the bus enters the data transfer state ( $T_d$ ) allowing external devices to drive the bus. The 80960SB then senses the READY\* signal to determine when data is available for a transfer. As long as READY\* is sensed asserted during the  $T_d$  state, data is transferred by the 80960SB. In this manner, the 80960SB can receive or transmit data each  $T_d$  state. Although this appears to provide a full speed throughput rate, the 80960SB only transfers up to 16 contiguous bytes that always occur on 16-byte even word address boundaries.

At the end of the final transaction for the burst operation, the 80960SB asserts the burst-last or BLAST• signal to signify the end of the burst transaction. The bus state control block then exits the data state and enters the recovery state ( $T_R$ ) while deasserting the bus enable signals. The  $T_R$  state provides a single bus cycle for external devices to recover from any previous access. As indicated in the bus state diagram in Figure 5-5, the bus can immediately enter a subsequent address state for another burst transaction. Thus the 80960SB microprocessor can transfer up to 16 bytes of data each ten clock cycles. For the Omniview system with a 20 MHz system clock frequency, this corresponds to a transfer rate of 16 million bytes per second.

Slower memory or peripheral devices can be supported by the 80960SB through use of the READY• control signal. At the end of any data transfer state  $T_d$ , the bus control logic checks the READY• logic. If the READY• signal is unasserted, the bus state control logic enters a wait state for the next cycle maintaining the status of the bus control signals. As long as the READY• signal is unasserted, the 80960SB will remain in the wait state indefinitely. The data/wait state is exited only when the READY• signal is asserted.

The 80960SB microprocessor supports single transfer read or write operations with the same burst-bus interface signals. A timing diagram of a single transfer read and a single transfer write operation is shown in Figure 5-7. The single transfer works the same way as the burst-bus operation except that the BLAST• signal is asserted early. In Figure 5-7, the read transfer is shown with a single wait state while the write cycle is shown with two wait states. The burst-bus example in Figure 5-6 was shown with zero wait states. The bus states are indicated at the top of the diagram in Figure 5-7 and show the normal flow of the 80960SB microprocessor bus control logic.

#### 5.4 CONTROL LOGIC IMPLEMENTATION

The control logic for the host processor design was implemented using programmable logic devices. All of the control logic for the host processor could be implemented easily in a single FPGA architecture device, however, it was decided early in the development to use discrete PLD devices for each major function in the control logic design to minimize design time.

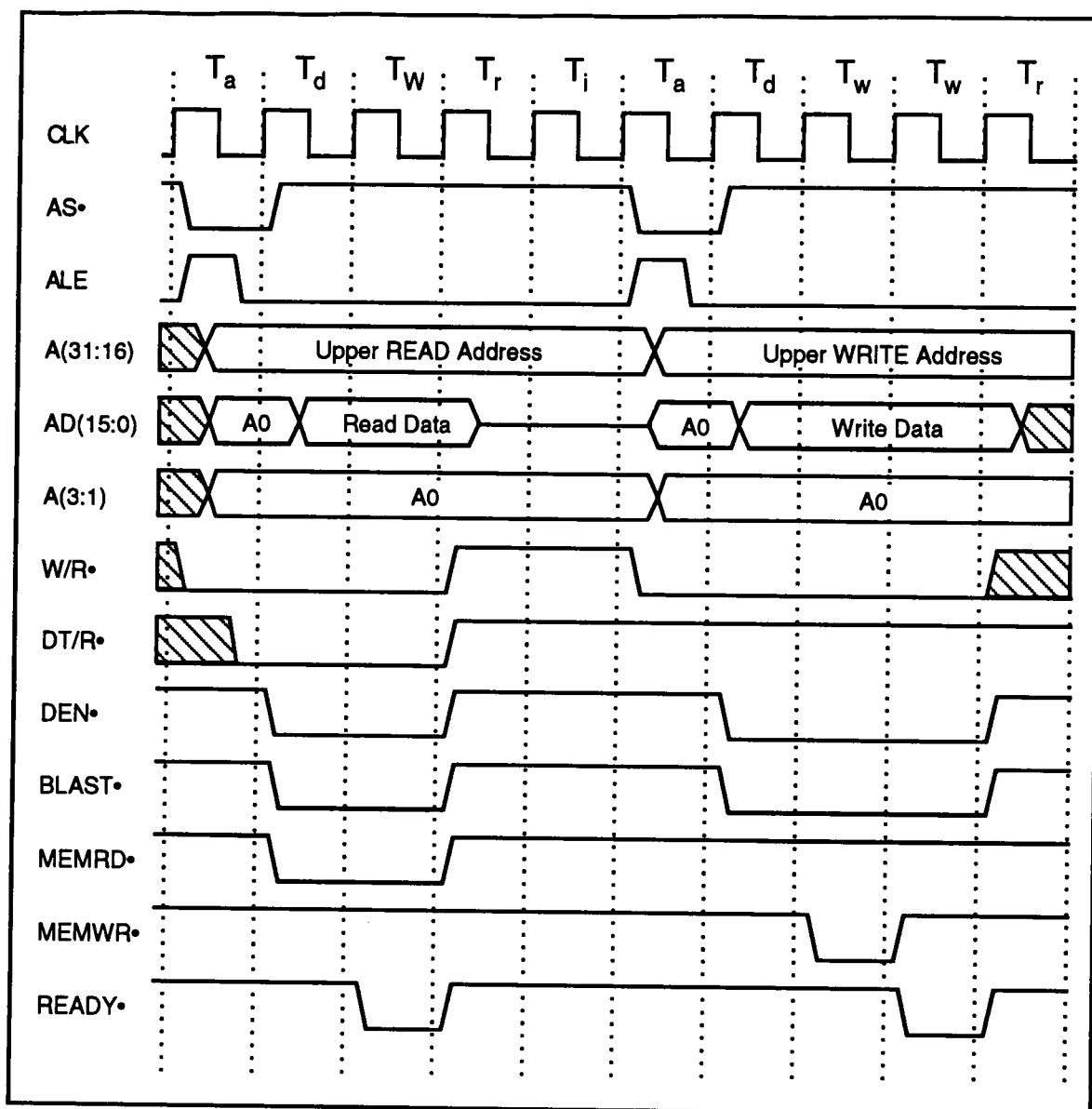


**Figure 5-7. Read/Write Timing Diagram (Non-Burst)**

The address decoding is accomplished with a single Lattice Inc. GAL18V10 programmable logic device which provides the major memory mapped address generation. The GAL18V10 functions as the main address decode and divides the memory space of the host microprocessor into several major function specific areas. The main memory space divisions are: the EPROM memory space, the SRAM memory space, the FLASH memory space, the peripheral I/O memory space, the input video memory frame buffer space, the output video display memory space, and non-functional or non-implemented memory

space. The memory map and base address for each of the main functional address spaces is shown in Figure 5-8.

The main address area for the host processor specific functions is located in the lower four megabytes of the microprocessor address space from hex address 0x00000000 to hex address 0x003FFFFFFF. The lower four megabytes is mapped into four evenly divided one megabyte memory areas for the memory and peripheral I/O control. The lower one-megabyte of address space is reserved for the EPROM or boot memory space. The next one-megabyte space above the boot memory space is reserved for SRAM based volatile memory storage. The third one-megabyte memory space from 0x00300000 to 0x003FFFFFFF hex is reserved for peripheral I/O functions.

The peripheral I/O memory space is further divided into eight functional address spaces. A 74ACT138 3-to-8 decoder is used to split the peripheral memory space into eight 128K memory spaces for each of the main peripheral control functions supported by the Omniview system. The peripheral and control functions for the Omniview system are divided into six functions: the input video memory control logic and arbiter, the output display video memory control logic and arbiter, the video processor board control space, the image processor board control logic, the real-time clock (RTC) interface, and the KIO peripheral I/O control logic. An extra select line from the decoder was reserved for additional expansion. Two of the control lines from the decoder were needed to support the KIO control and interrupt vector interfacing.

### **BUS Tracker Control Logic Implementation**

The read/write control logic for the host processor is implemented using three Lattice Inc. GAL22V10 programmable logic devices (PLDs). Each GAL22V10 control PLD provides the bus tracking, read/write, and ready control logic for the main functional control blocks - the system memory, the input and display video frame buffer control, and the peripheral I/O and control functions.

Each of the control PLDs is designed to implement the bus tracking, the read/write generation, and the ready and wait state signal generation in order to interface the 80960 to external devices. Since each of the main functional control PLDs are implemented in a similar manner, only the memory control PLD that controls the interface to the system memory is discussed.

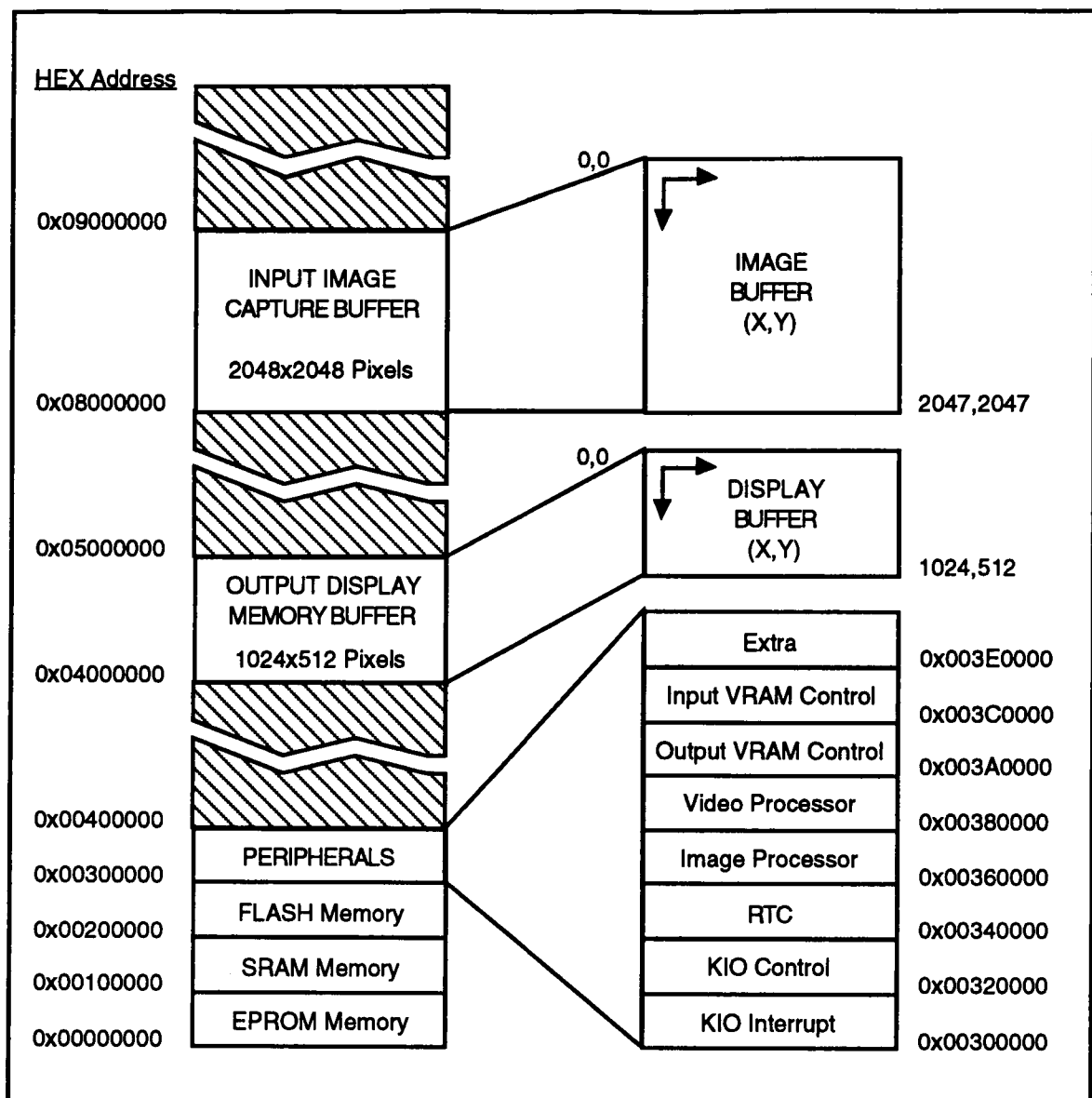


**Figure 5-8. Host Processor Memory Map**

The main memory is implemented using three types of memory - EPROM, SRAM, and FLASH memory. The main memory address space allows both read and write accesses to each of the three types of memory. The main memory control PLD provides the bus tracker interface and ready wait state generation to interface the 80960 microprocessor to the system memory.

The memory control PLD tracks the burst bus signal clocking of the 80960SB microprocessor in order to support both single read/write accesses and burst mode

read/write accesses. Another function of the memory control PLD is to generate wait states to support slower EPROM and FLASH memory devices. The memory control PLD decodes the 80960 bus signals and determines the type of memory cycle, generates the appropriate control signals for the system memory, and provides wait state generation. The memory control PLD decodes the following control signals from the microprocessor - AS•, W/R•, BE1•, BE0•, and BLAST•. The device select signals, EPROM•, SRAM•, FLASH•, and NONSEL•, are sampled by the memory control PLD to determine which memory space is being accessed. The memory control PLD provides three sets of control signals for each of the memory interfaces: OE•, WRH•, and WRL•. Each type of memory has an individual OE• signal which functions as the output drive enable signal. A separate set of WRH• and WRL• signals are provided for the SRAM and FLASH memory and provide the write enable signals. The WRH• enable signal selects the upper byte while the WRL• signal selects the lower byte of the data bus for write operations. The write enable signals are not supported for the boot memory space since it implemented with read only devices. Finally, the memory control PLD generates wait states for memory accesses in order to allow slower EPROM and FLASH memory devices to be used in the design.

The basic state machine diagram for the memory control PLD is shown in Figure 5-9. The state diagram shows the state transitions for the FLASH memory control only. The EPROM and SRAM control state machines are similar. The function of the bus tracking state machine is to decode the bus signals to determine the type of access and whether a burst bus cycle or a single read/write cycle is being generated. As discussed in previously, the 80960 can support burst type bus accesses transferring up to 32 bytes per burst cycle. During a burst bus cycle, the bus can transfer data at a rate of up to 1/2 the microprocessor clock. In order to support slower memory devices, a ready signal is generated by the memory control PLD so that each type of memory in the system can have different wait states. The state diagram in Figure 5-9 shows the FLASH memory control state machine and the FLASH memory read/write state diagram. The state machine for the FLASH memory implements a 2-2-2-2-2-2-2-2 wait state type access for the FLASH memory during the eight cycle burst. There are two wait states for each read or write access.

The state machine detects the address state when AS• and the memory select or FLASH• signals are both low. The read or write condition is determined by the W/R• signal from the processor and determines the appropriate branch condition from idle. When the state machine transitions to the RDY state it asserts the READY• signal.

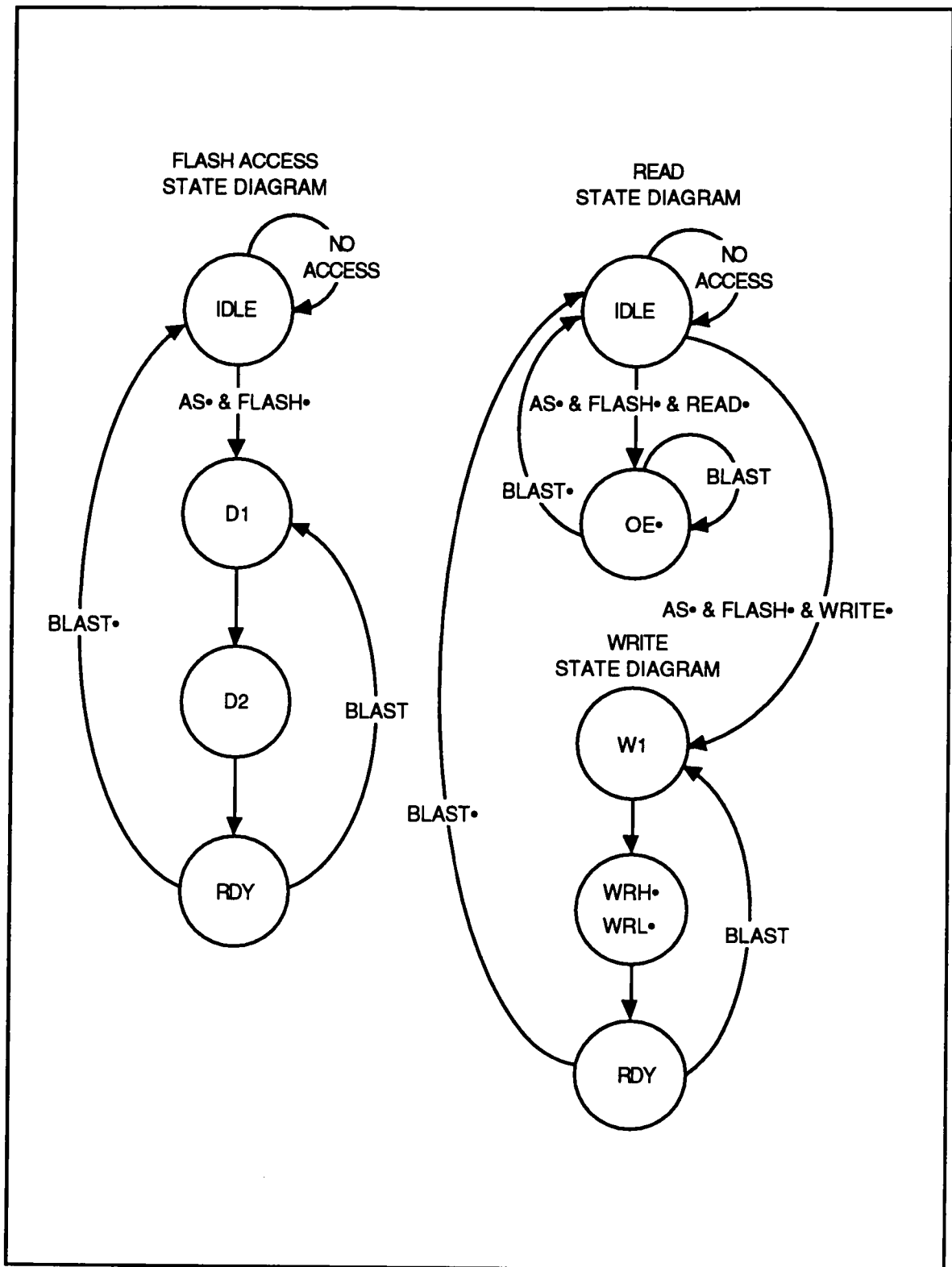


**Figure 5-9. Bus Tracker Control PLD State Diagram**

During the RDY state, the state machine senses the BLAST• signal from the processor to determine whether to continue with the next data transfer in the burst. If BLAST• is asserted, the transfer is finished and the state machine transitions back to the idle state condition while the processor bus control state machine transitions to the recovery state. The bus tracker state diagram will therefore support both single-cycle accesses or burst-mode accesses to the FLASH memory.

The 80960SB's burst bus interface requires that the main memory implementation use full word wide or 16-bit wide memory accesses in order to insure maximum performance. As discussed previously, the burst bus access implementation provides the capability for the 80960SB to burst read or write up to 32 bytes or eight 16-bit words of data during a single bus operation.

## **5.5 MEMORY IMPLEMENTATION**

There are three types of memory used to implement the Omniview host processor system. Most microprocessor systems use standard SRAM and EPROM for the main memory. Some larger systems also use DRAM because of its higher density and lower cost over static memory or SRAM implementations. The primary goal of the Omniview design is to provide a high performance, stand-alone embedded system that can be integrated into user applications. In order to provide the best performance versus cost implementation, SRAM was chosen for the main volatile memory implementation. Static RAMs have much higher bus performance than dynamic memory and the control logic is much simpler. The SRAM memory was implemented using four 256K by 4-bit wide devices giving a total of 512K bytes of volatile memory for parameter and data storage. The SRAM memory supports both byte and 16-bit word read/write accesses with burst mode support.

The main operating system and boot program is implemented using two 8-bit wide EPROMs. The host processor design uses two 32-pin EPROM sockets that can support several different memory size devices. The initial memory map implementation supports up to two 512K by 8-bit wide EPROM devices. This provides up to one megabyte of operating system memory support. The 32-pin sockets were configured so that smaller memory device densities can be used to support different sized operating system kernels.

A jumper programmable option determines the size of the EPROM memory devices that are being used.

## **FLASH Memory Support**

The Omniview system uses FLASH memory technology for application programming and operating system development. The recent advances in FLASH memory technology has brought about devices with densities equal to or greater than EPROM memory devices and access speeds below 100 nS. FLASH memory provides significant advantages for an embedded design such as the Omniview system. The main advantage that FLASH has to offer is that it can be erased and programmed many times while installed in the circuit board. FLASH memory erase-program cycling is guaranteed up to 10000 erase-program cycles while newer devices are approaching 100,000 erase-program cycles.

The Omniview system provides two 32-pin PLCC sockets that can hold up to two 512K by 8-bit FLASH memory devices providing up to 1024K bytes or 1 megabyte of on-board re-programmable application memory. The FLASH memory is used to accomplish two main functions. First, the FLASH is used to speed development of new operating system kernels for the Omniview system. The memory mapping control logic was designed to allow the FLASH memory space to replace the EPROM boot memory space during operation. Using the FLASH memory for testing new operating system kernels significantly reduces the software development time since new operating system kernels can be programmed and tested with a few minutes. The second main function of the FLASH memory is to provide the ability to upload new application programs for the Omniview system's normal operation. The application programming is discussed further in Chapter 7.

## **5.6 PERIPHERAL AND I/O INTERFACES**

Almost all host microprocessor systems need some sort of serial and parallel input/output capabilities to support interfacing both to the real world and to handle on-board special purpose functions. As mentioned previously, one of the main decisions for the Omniview system is reliability, cost, and performance. Although most of the functions of parallel and interrupt input/output functions can be implemented fairly simply using standard discrete components, the desire for the Omniview system is to use a highly

integrated component to handle as many of the general and special purpose I/O functions as possible.

The Zilog Z84C90 CMOS (KIO) peripheral I/O device was selected to implement all of the I/O needs for the Omniview host processor board. The Z84C90 provides a large degree of VLSI integration to incorporate just about all of the I/O functions one would need for general interfacing needs. The Z84C90 KIO is a multi-channel, multi-purpose I/O device contained in an 84-pin plastic PLCC packaged integrated circuit. Designed in CMOS technology for low power operation, the KIO combines the features of several industry standard peripheral components in one device. The Z84C90 combines the features of the Zilog Z84C30 Counter Timer Chip or CTC ( a 24 pin device), the Zilog Z84C40 serial I/O or SIO ( a 40-pin device), a Z84C20 parallel I/O or PIO ( a 40-pin device), an on-chip oscillator circuit, and a byte-wide bit-programmable I/O port or PIA into a single 84-pin PLCC device. A functional block diagram of the Z84C90 KIO is shown in Figure 5-10.

The host programming and control interface to the Z84C90 KIO consists of sixteen byte interface read/write registers. A subset of these registers are allocated for and used to control each of the internal functional blocks of the KIO. Two of the control registers are used to program overall control functions for the KIO device such as interrupt daisy chaining and internal resets. A brief description of each of the KIO sections is provided in the following sub-sections.

### **KIO Parallel I/O Block**

The Z84C20 parallel I/O or PIO block provides both TTL and CMOS parallel interfaces between peripheral devices and the host microprocessor through two 8-bit programmable parallel ports. The parallel ports, designated as Port A and Port B, are byte-wide ports and have several modes of operation. Each port can be programmed as input, output, or bi-directional.

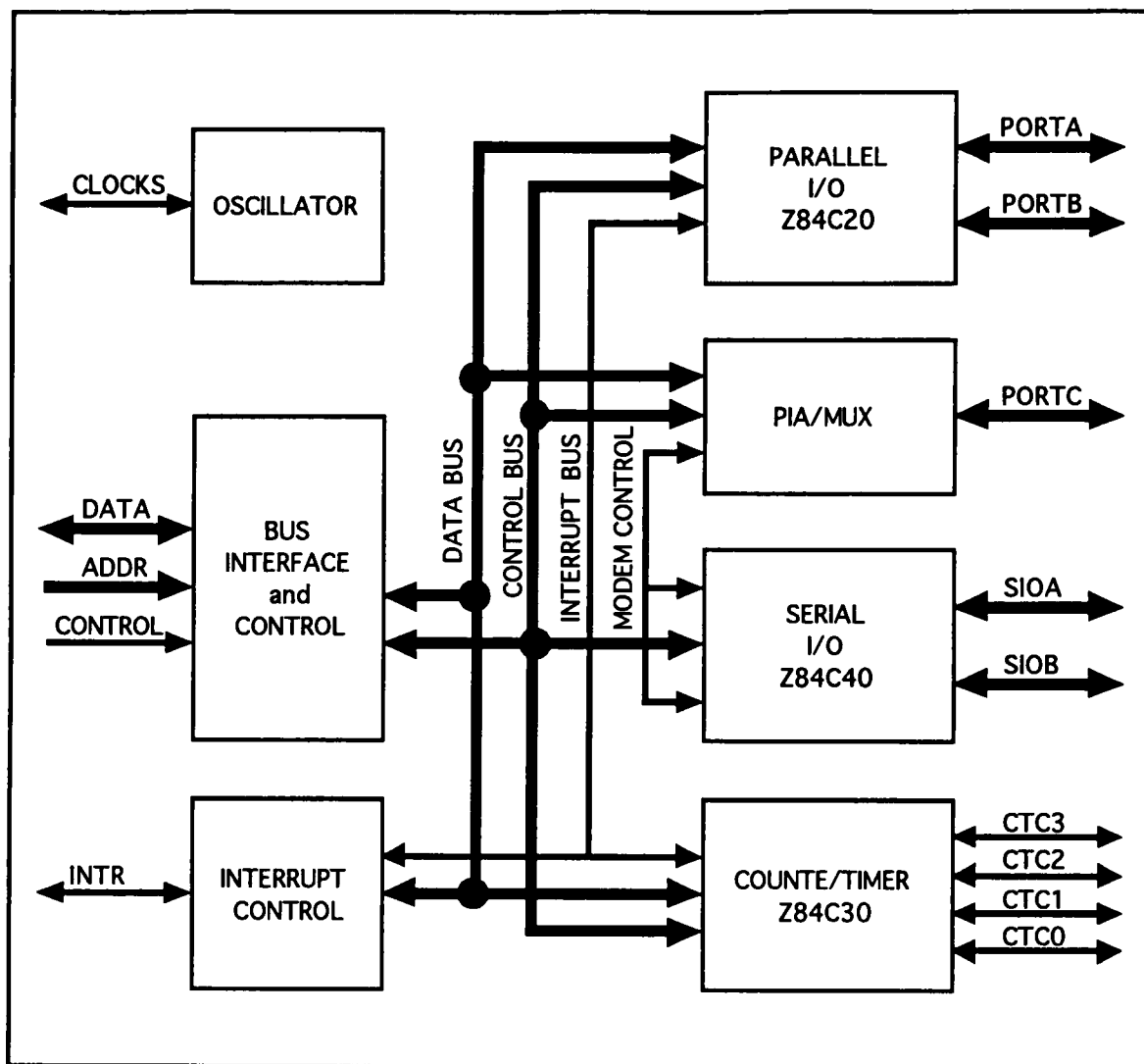


**Figure 5-10. KIO Functional Block Diagram**

The Port B section is used in the Omniview system as an output to control various enable and reset functions. For instance, the master resets for the Image processor and Video Processor boards is controlled by different Port B bits. An additional feature of the PIO interface block is that the PIO ports A or B can also be programmed to generate interrupts to the host processor through the interrupt control block. The interrupt mode can be programmed to interrupt if the inputs are either logically high or low. They also can be programmed to either logically OR or AND the input conditions. This mode is useful to enable detection of multiple assertion of interrupts. For the Omniview system, Port A was programmed as an 8-bit input with active-low interrupt conditions ORed so that when any

of the bits becomes asserted the KIO signals an interrupt to the host. The Omniview system uses the Port A interrupts to detect vertical syncs from the video genlock logic, to determine when the denominator processor is finished processing, and to implement a timer using the RTC.

### **KIO Counter/Timer Block**

The Counter Timer or CTC block of the Z84C90 integrates four individual 8-bit counter/timer channels that can be programmed to act as either timers or counters. These counter/timers can be programmed independently and can also be chained to provide up to a 32-bit counter or timer with an input frequency from DC to 10 MHz. Each of the CTC channels, designated as channels 0 through 3, have an 8-bit pre-scale as well as its own 8-bit counter to provide a wide range of resolution. Each of the four channels has individual clock and trigger inputs to count input pulses and to indicate a zero crossing time-out. Each channel can also generate an independent interrupt vector for the host. The Omniview system uses two of the counter timers to generate baud clocks for the SIO portion of the KIO. The baud clocks provide the capability to provide independent serial baud rate communication modes for each of the two serial ports. The other two CTC channels were used to provide a general purpose timer for event timing. The event timer was useful for measuring the programming pulse width for programming the FLASH memory devices installed on the host processor board.

### **KIO Serial I/O Block**

The serial I/O or SIO unit provides two complete serial I/O channels designated as Channel A and Channel B. The two serial I/O channels provide the common parallel-to-serial and serial-to-parallel conversions and can be programmed for a number of serial communications applications. Both channels can be programmed to provide common asynchronous and synchronous byte-oriented or bit-oriented protocols such as SDLC and HDLC. Each channel has separate transmit and receive pins and can support full-duplex communications. The Omniview system uses both ports with external RS-485 or RS-422 compatible drivers and receivers to implement external communications and control ports. The RS-422 drivers used with the SIO block provides the capability to communicate at high data rates over low-capacitance twisted-pair cables at distances up to 1000 feet if necessary.

## **Real-Time Clock Support**

Another device included in the design of the Omniview host processor core is the DS1284 Real-Time Clock (RTC) from Dallas Semiconductor Corp. The DS1284 RTC provides a 24-hour clock calendar with a battery and crystal interface in a 28-pin PLCC package. The RTC provides the ability to generate alarms and alarm interrupts as well as maintain time and calendar information. It also contains 50 bytes of battery-backed static memory. The RTC can be used to generate sequence timing for control of the viewing angles of the Omniview system. The integrated battery backed memory is also valuable for maintaining operating system status and configuration data. For instance, in a security application the configuration data might indicate a particular sequencing of the views for monitoring specific areas. Also, the RTC static memory can be used for setup parameters to define field-of-view, image center offsets in the genlock interface, and other parameters as necessary.

## **5.7 SUMMARY**

The host processor core design was designed around the Intel 80960SB microprocessor to provide real-time control of the Omniview camera system. The 80960SB microprocessor is a 32-bit RISC type microprocessor with several enhancements that promote real-time performance. The bus control interface supports burst mode data transfers that allow data to be transferred at 1/2 the processor system clock frequency. A complete floating point execution unit provides IEEE-796 80-bit hardware floating point operation. Finally, the 80960 architecture uses register scoreboarding and parallel execution to further enhance its performance.

The memory design for the host processor uses two approaches that support and enhance the development of embedded, stand-alone operation. The host uses EPROM memory for the boot program and main operating system. A bank of static SRAM memory provides the volatile parameter and variable storage for the system. The system uses FLASH memory to provide storage and update of the application program. The application program can be added on top of the boot and operating system and provides the ability for users to upgrade the application software in the field. Finally, the address decode logic provides the ability to swap the EPROM and FLASH memory spaces so that new operating

systems can be developed and tested without having to erase and burn the existing EPROM data.

The peripheral I/O control for the host processor was implemented using the Zilog Z84C90 KIO device. The KIO contains the capabilities of several peripheral I/O functions in a single highly integrated 84-pin PLCC package. The KIO provides programmable parallel I/O, two serial UARTS, four clock-timer functions, and interrupt processing for the host processor core. A real-time clock circuit (RTC) with battery backup was included in the design to provide a time-of-day and clock calendar alarm function. The RTC device also provides additional battery backup memory for storage of system setup parameters.

## CHAPTER 6

# OMNIVIEW VIDEO PROCESSOR DESIGN

The Omniview prototype system was designed so that each major processor board could be modified without impacting the design of the other processors. The Omniview video processor board was designed so that all of the video processing functions are contained on a single board. A functional block diagram of the video processor board is shown in Figure 6-1. The video processor board contains five main functional blocks: the video genlock and digitizer interface, the input video frame buffer memory, the interpolator filter block, the output display frame buffer, and the video output display block.

The video board contains two 60-pin connectors that interface to the host processor board. One of the connectors provides the host processor address, data, and control signal interface. The second connector provides the video memory address, data, and control signals. A memory map of the video processor board is shown in Figure 6-2. The video board contains seven main functional control blocks as shown in memory map. Each control block shares the same amount of mapped address space. An eighth memory space area is reserved for future expansion. The programmable control functions for the video processor board are: the video A/D digitizer, the horizontal line lock and VCO controller, the genlock video sync generator, the output display sync generator, the interpolator coefficient look-up-table, and the interpolator filter scale register. A control register is included to control overall reset and synchronization of the video board functions.

The input genlock and output display video memory buffers are also located on the video processor board. The input video memory buffer is implemented in four banks of 512 by 2048 horizontal pixels. The video digitizer can therefore capture a video image up to 2048 by 2048 pixels. The output display memory buffer is organized as a 512 by 1024 pixel buffer. The output display generator uses 480 vertical by 640 horizontal pixel resolution to support the standard square pixel NTSC format. Although the input and output video memory arbitration and control logic is physically contained on the host processor board, the implementation is included in this chapter.

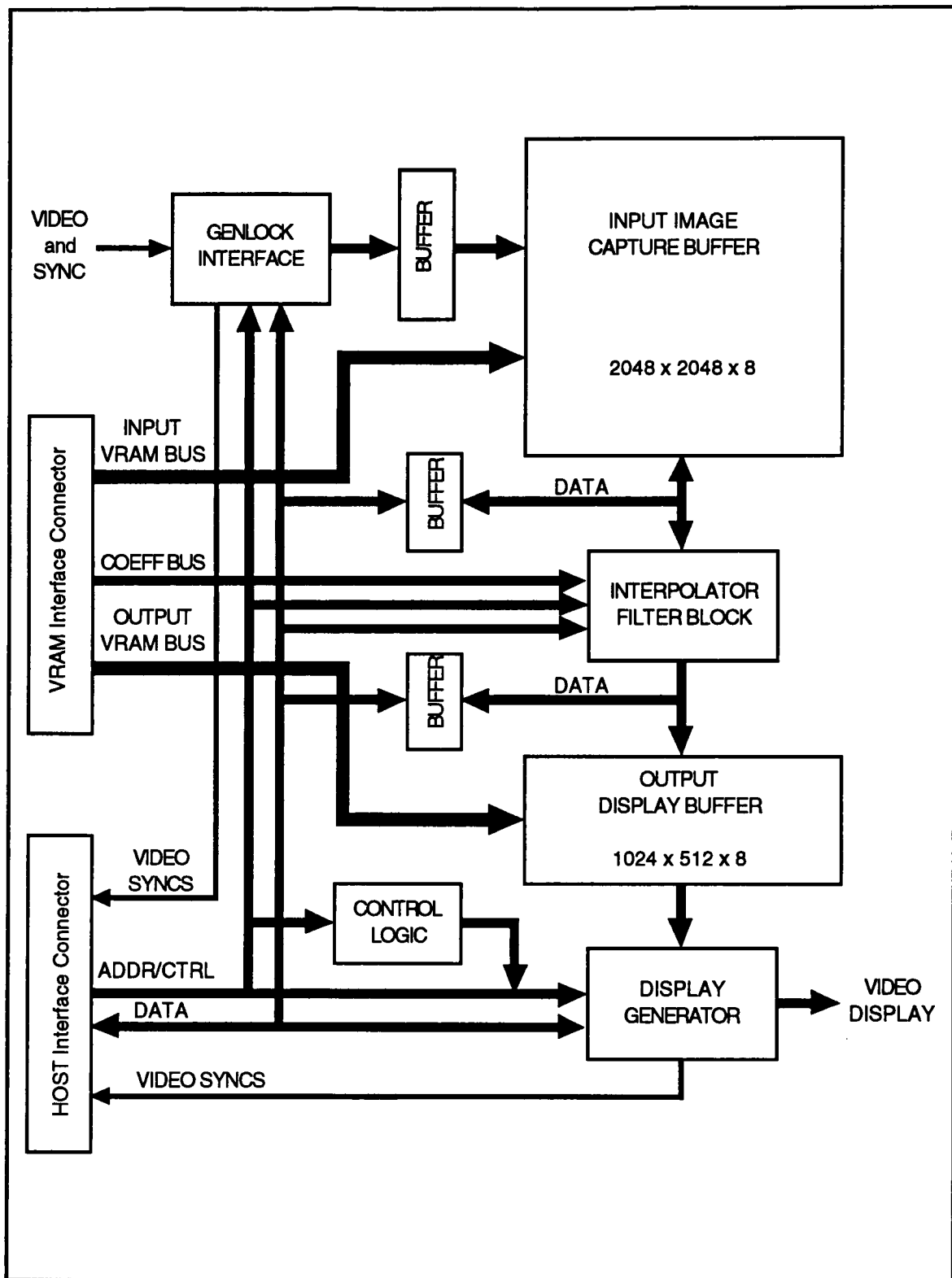


**Figure 6-1. Video Processor Board Functional Block Diagram**

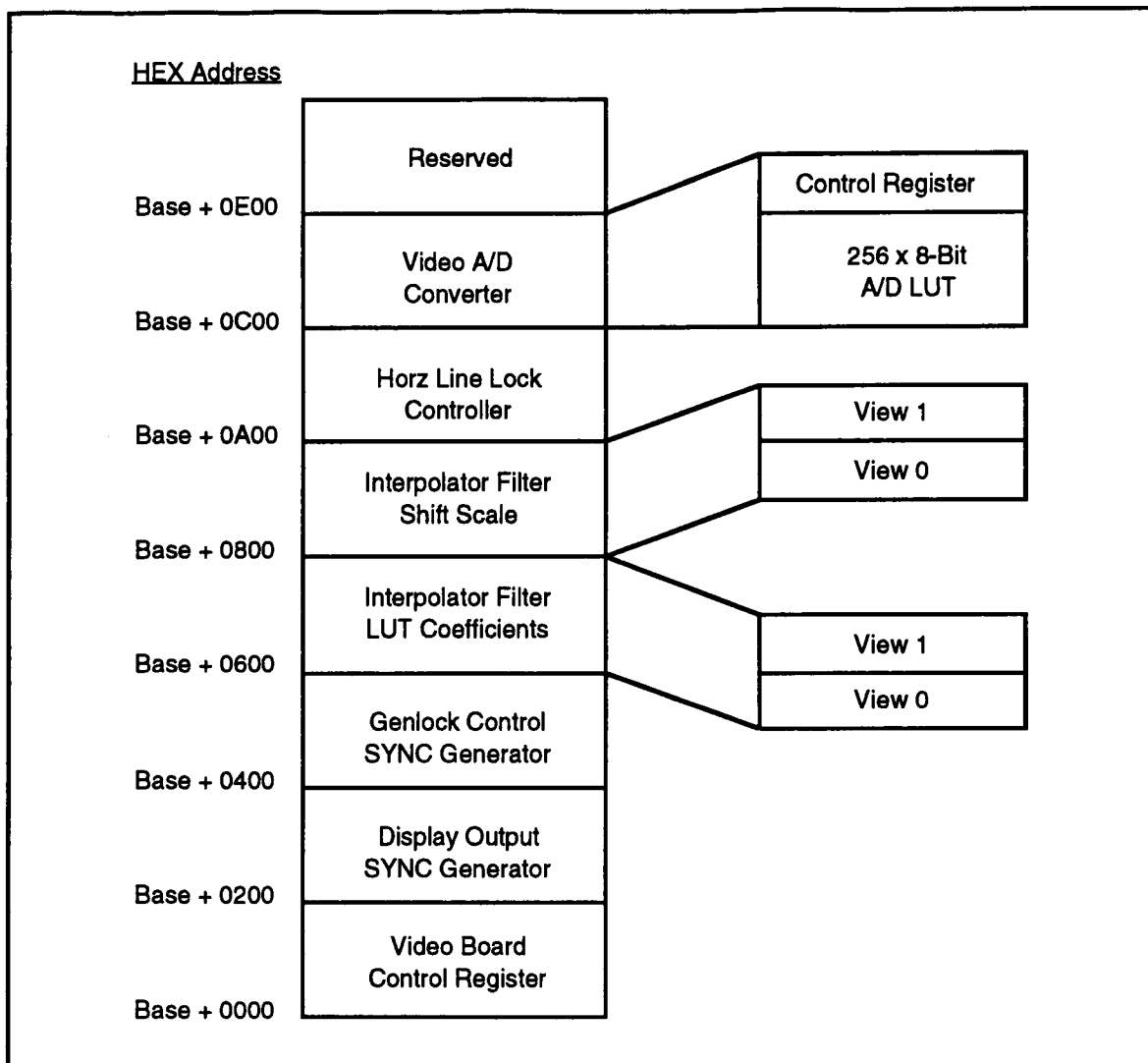


**Figure 6-2. Video Processor Board Memory Map**

## 6.1 VRAM CONTROLLER AND MULTI-PORT ARBITER

The input and output video memory buffer control logic interfaces are physically located on the host processor board. The video memory interface logic provides all of the control, video scan synchronization, and arbitration for each video frame buffer. Since the implementation of the input and output video memory buffer interfaces are very similar, only the input video memory control logic is discussed in this section. The video memory interfaces were implemented using a hybrid approach in order to meet the best tradeoff between performance, implementation, and project deadline requirements. Future versions

of the Omniview design will provide faster video memory accesses using higher speed logic.

The video memory buffer interfaced was implemented using a National semiconductor VRAM memory driver device and an Actel FPGA. A functional block diagram of the video buffer interface is shown in Figure 6-3. The input memory consists of four banks of video memory organized as a 2048 by 2048 pixels. The DP8521 device provides the address multiplexing and control synchronization for the VRAM devices. It contains row and column address latches with high speed, high current buffers that can drive large VRAM memory arrays. The DP8521 also provides the multiplexed address to the VRAM devices from the row and column address present on the input. Internal circuitry within the DP8521 controls refresh generation and guarantees critical timing of the VRAM control signals. The most critical timing for VRAM devices is the row and column address multiplexing and the row address pre-charge times. The DP8521 makes sure that each bank of video memory has sufficient row address precharge time before each access takes place.

A single Actel FPGA is used to control the DP8521, to provide memory bank control, and to provide the arbitration for the video memory. The FPGA based video system controller (VSC) provides the arbitration and address multiplexing between the image processor, the host processor and the video update control block. The VSC contains a prioritized arbitration controller that allows each section to access the video memory in the most efficient manner.

A functional block diagram of the VSC is shown in Figure 6-4. The VSC provides the control and interface functions and uses the DP8521 as the video memory address driver and VRAM control signal generator. The VSC contains an internal video memory controller that detects the horizontal and vertical sync signals from either the input genlock digitizer block or the output graphics display generator. The video transfer control logic is responsible for the synchronous updates of the video memory to and from the video DRAM array. During each horizontal active video interval, video data is clocked into the video memory devices. During the horizontal sync interval, each line of video data is transferred to a row of memory in the video buffer.

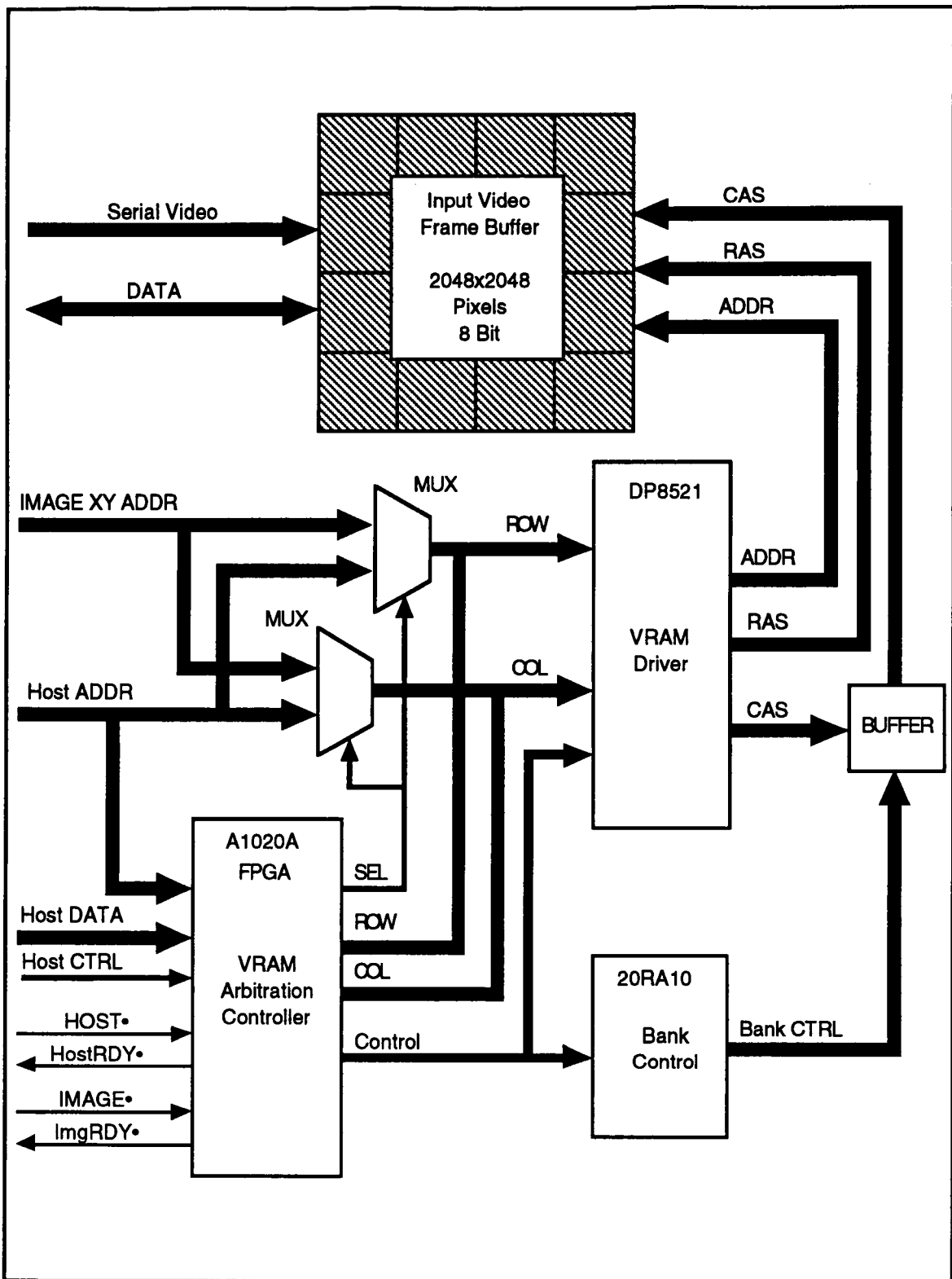


**Figure 6-3. Input Video Memory Buffer Interface Block Diagram**

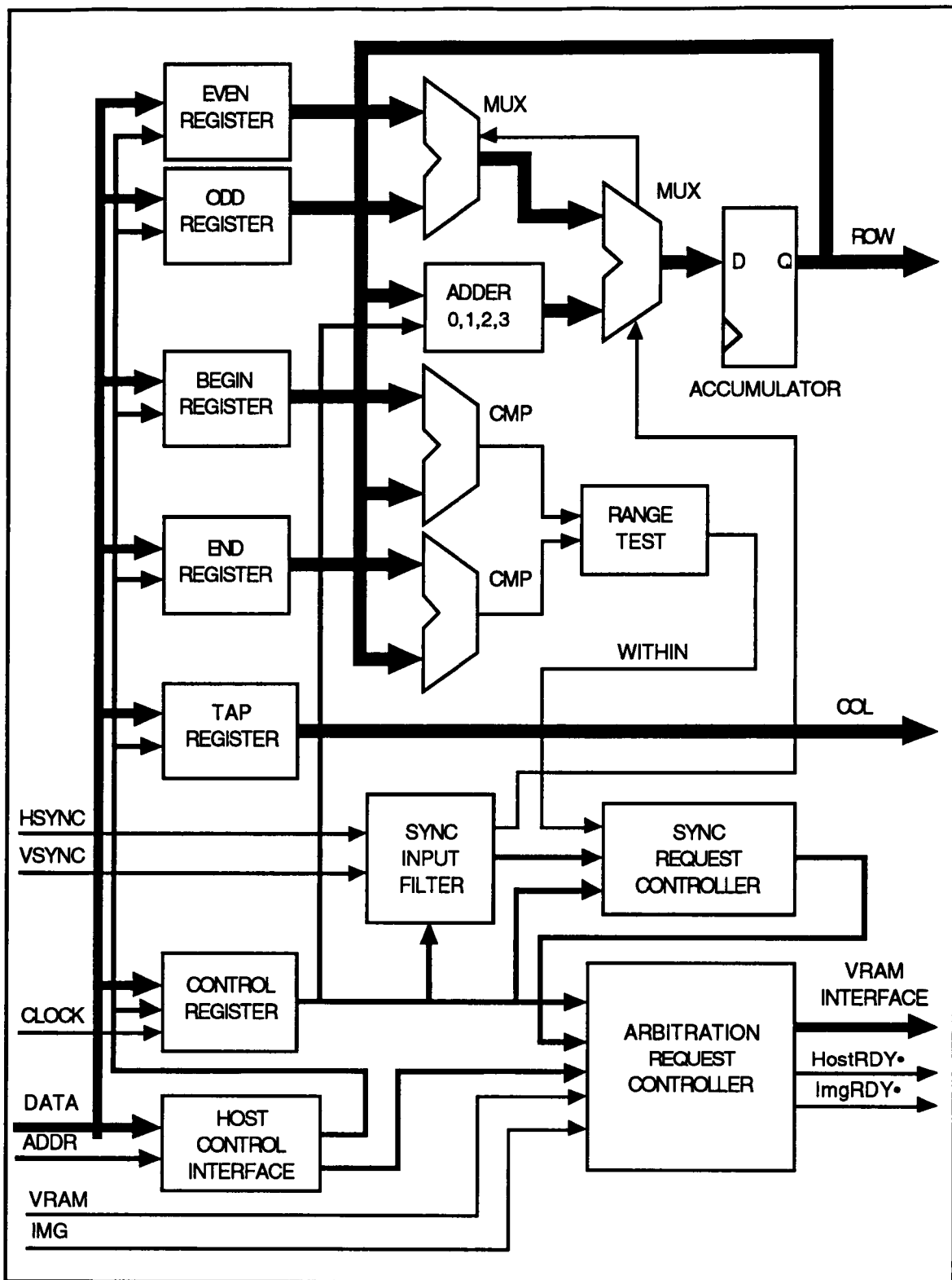


**Figure 6-4. Input VRAM Arbitration and SYNC Controller**

The VSC maintains a row counter that is incremented after each video line transfer. The vertical sync signal tells the VSC when to reset to the top row of the video frame buffer to begin capturing a new video frame. The arbiter insures that the VSC internal video transfer logic has the highest priority access to the video memory. This insures that video scan line updates remain synchronous with the horizontal and vertical synchronization intervals.

The VSC arbiter also provides address multiplexing and arbitration between the image processor and the host processor. To accomplish the address multiplexing, external 2-to-1 high speed multiplexors with tri-state enables are used to select the currently active port. The 2-to-1 multiplexors select either the image processor address or the host processor address for the next VRAM read/write operation. The tri-state enable allows the multiplexors to be disabled when the internal VSC row and column address are driving the DP8521 address inputs during VRAM transfer cycles. The VSC arbiter provides separate access and ready signals to each of the control blocks. The video arbitration control approach allows each of the video memory access blocks to run independently and asynchronously to each other. The other function that the DP8521 and VSC implementation provides is external refresh generation between accesses at a 15  $\mu$ S interval. The 15 $\mu$ S refresh rate ensures that each row of the video memory is refreshed within an 8 mS interval. The arbitration scheme also ensures that back-to-back image processor accesses to the video memory is the least delayed between arbitration requests.

## **6.2 VIDEO GENLOCK INTERFACE**

The video board contains a fairly general purpose genlock interface that can digitize video to 8-bit gray scale values. The video genlock system uses a horizontal line locked VCO implementation. The genlock approach allows the Omniview video interface to synchronize to RS-170 standard level video signals. A simplified functional block diagram of the genlock digitizer interface is shown in Figure 6-5. The digitizer is implemented using a single A/D converter and a phase lock VCO circuit to synchronize to the video signal. The genlock circuit also contains an external video sync generator which is not shown in Figure 6-5. The video sync generator is provide to support external video cameras that require either standard or non-standard vertical and horizontal synchronization signals.

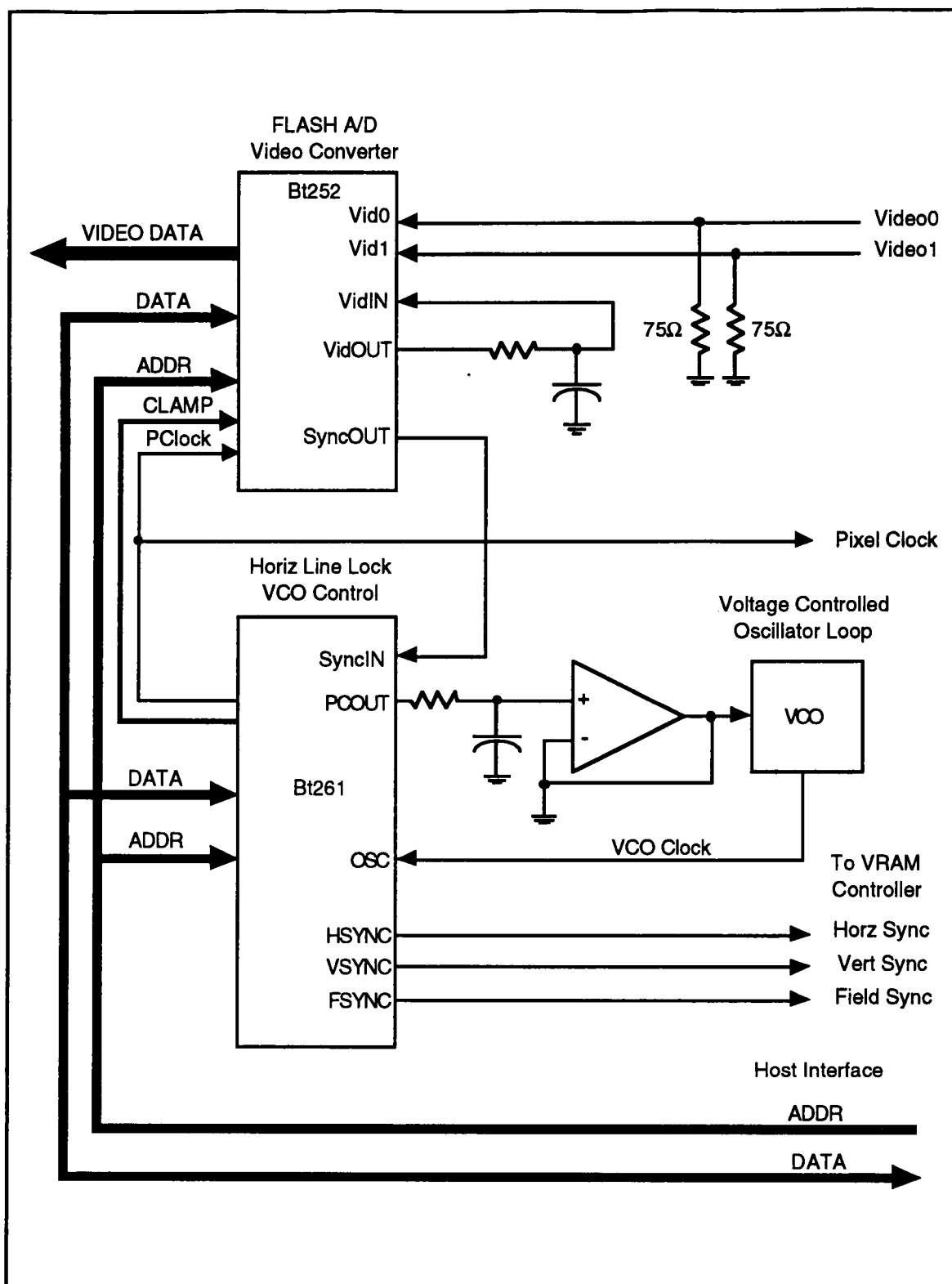


Figure 6-5. Video Genlock Interface Block Diagram

## **Video Digitizer Discussion**

The video digitizer was implemented using a Brooktree Corp. Bt252 video A/D converter. The Bt252 was used because it met the requirements for the delivery of the first prototype Omniview system and provide the capability to digitize standard RS-170 level video signals with 8-bit resolution up to a frequency of 30 MHz. The Bt252 is designed to digitize standard video signals such as NTSC and CCIR. The digitizing rates are more than sufficient to support the standard resolution cameras using 640 by 480 square pixel NTSC quality.

The Bt252 contains a 4-to-1 input multiplexor that feeds the internal A/D converter. An internal sync detect circuit allows contains an internal 4-to-1 multiplexor that allows either video input to be used to generate sync information for the genlock circuit. The detected sync is output using TTL levels for external digital phase lock loop applications. The output of the video mulitplexor is digitized using an 8-bit flash A/D converter. The output of the A/D converter addresses an internal 256 by 8-bit lookup table enabling real-time image manipulation prior to video memory storage. The input video LUT offers the capability of including contrast enhancement, thresholding, reverse video, or implementing other non-linear video effects.

At 30 MHz bandwidth, the Bt252 will support a 1050 line non-interlaced camera at up to 15 frames-per-second scan rate. This was important since one of the cameras used in the delivery of the prototype system was a special JVC camera that provides the 1050 lines vertical lines of resolution at a frame rate of 15 Hz. The JVC TKF7100 camera, shown in the photograph in Figure 8-5, uses a high resolution Saticon™ tube with slow scan to provide higher resolution. The tube is scanned sequentially using a 4/3 aspect ratio which provides double the resolution of the standard NTSC images at one-half the vertical frame rate. The Omniview system digitizes the JVC TKF7100 camera at a resolution of 1000 vertical by 1280 horizontal pixels using a 24.54 MHz center frequency VCO.

### **Horizontal Line Lock Controller**

The Omniview genlock interface uses a Brooktree Bt261 line lock controller (LLC) to implement a VCO based phase-locked loop for digitizing the input video. The Bt261 provides the capability to implement a phase lock loop that can synchronize to the horizontal sync signals and provide the video pixel clock to drive the A/D converter.

The LLC contains an internal sync detect and separator that recovers horizontal and vertical sync information from an video signal. Either composite video or TTL composite sync information is sampled by the LLC. The LLC internal 12-bit registers that allow programmable control of horizontal and video timing so that standard and non-standard video sync timing information may be recovered.

The Omniview LLC contains an internal phase comparator that can be used with an external VCO in order to provide a pixel clock that is synchronized to the horizontal sync frequency. The implementation supports active pixel sampling that is a multiple of the horizontal sync frequency. The Omniview system provides a socket that allows different VCOs to be used to support cameras with non-standard syncs. The Omniview prototype system uses a 24.54 MHz VCO to allow both the JVC TKF7100 1050 line non-interlaced camera or a standard 525 line interlaced camera to be digitized. The LLC contains an internal divider that allows the VCO frequency to be divided down before sync timing is recovered. Therefore, the 24.54 MHz frequency can be used for the TKF7100 camera while a 12.27 MHz center frequency can be obtained for standard cameras.

The Bt261 LLC also contains internal noise gating that can be used to maintain the best line lock conditions. Noise gating allows the LLC to ignore extra sync pulses from disrupting the phase lock during the vertical sync interval. This allows the LLC to lock to video signals that have either single or double serration pulses during the vertical sync interval. The LLC supplies the horizontal and vertical sync signals to the input video memory control block. The horizontal and vertical syncs recovered by the LLC are used to determine when to update video data into the video frame buffer. During the active video signal, the video signal is sampled and pixel data is scanned into the video memory devices. During the horizontal sync interval, the video memory controller FPGA, discussed in section 6.1, performs a video transfer to write the entire line of video data into the correct row of the video memory bank. The video memory controller continues to write lines of video data until the vertical sync signal is detected. The VSC then resets the row counter to the top of the video frame buffer starting row. The genlock and VSC implementation allows video frame captures from cameras with up to 2048 lines and 2048 horizontal pixels of video data. The current implementation of the A/D converter interface, however, only supports video digitization at up to 30 MHz. A faster video genlock and A/D converter is being designed for future versions of the Omniview system.

### 6.3 VIDEO DISPLAY INTERFACE

The video display circuit provides up to a 1024 by 512 pixel resolution display and can be programmed to support several different frame rates and resolutions. The output display video frame buffer is implemented using four 1 Megabit VRAMs and provides a 1024 by 512 pixel resolution with 8 bit gray scale resolution. The output display buffer is implemented using an odd and even pixel interleaved format. The interleaved design provides the simplest implementation for the video line transfer updates.

The pixel sample rate is generated using a single oscillator which determines the pixel resolution. The pixel clock drives the video sync generator, the video sync PAL and the video display DAC. The video output is generated using a Brooktree Bt102 video DAC. The Bt206 provides synchronous input of the sync and blank signals and an 8-bit pixel value. The output of the Bt206 is a current source DAC capable of driving a 75 ohm load. The pixel clock rate of the video DAC can operate at frequencies up to 75 MHz providing the capability to support all of the standard video display formats including a 60 Hz non-interlaced 640 by 480 resolution display.

The output display generator provides the ability to generate several different synchronization standards. The video synchronization is controlled using a single chip video sync generator device. The National Semiconductor LM1882 provides all of the video synchronization clocking. The device requires a pixel clock input and provides complete programmable control of the video sync signals. Both interlace and non-interlaced sync signals may be generated. The width and timing of the horizontal and vertical sync signals may also be programmed for any value. The LM1882 also provides a separate set of sync signals that are programmable. The separate sync signals were used to generate the horizontal and vertical sync signals for the output display video system control logic. A single 20-pin 16R6 type PAL was used to control the sync signals that drive the video display DAC.

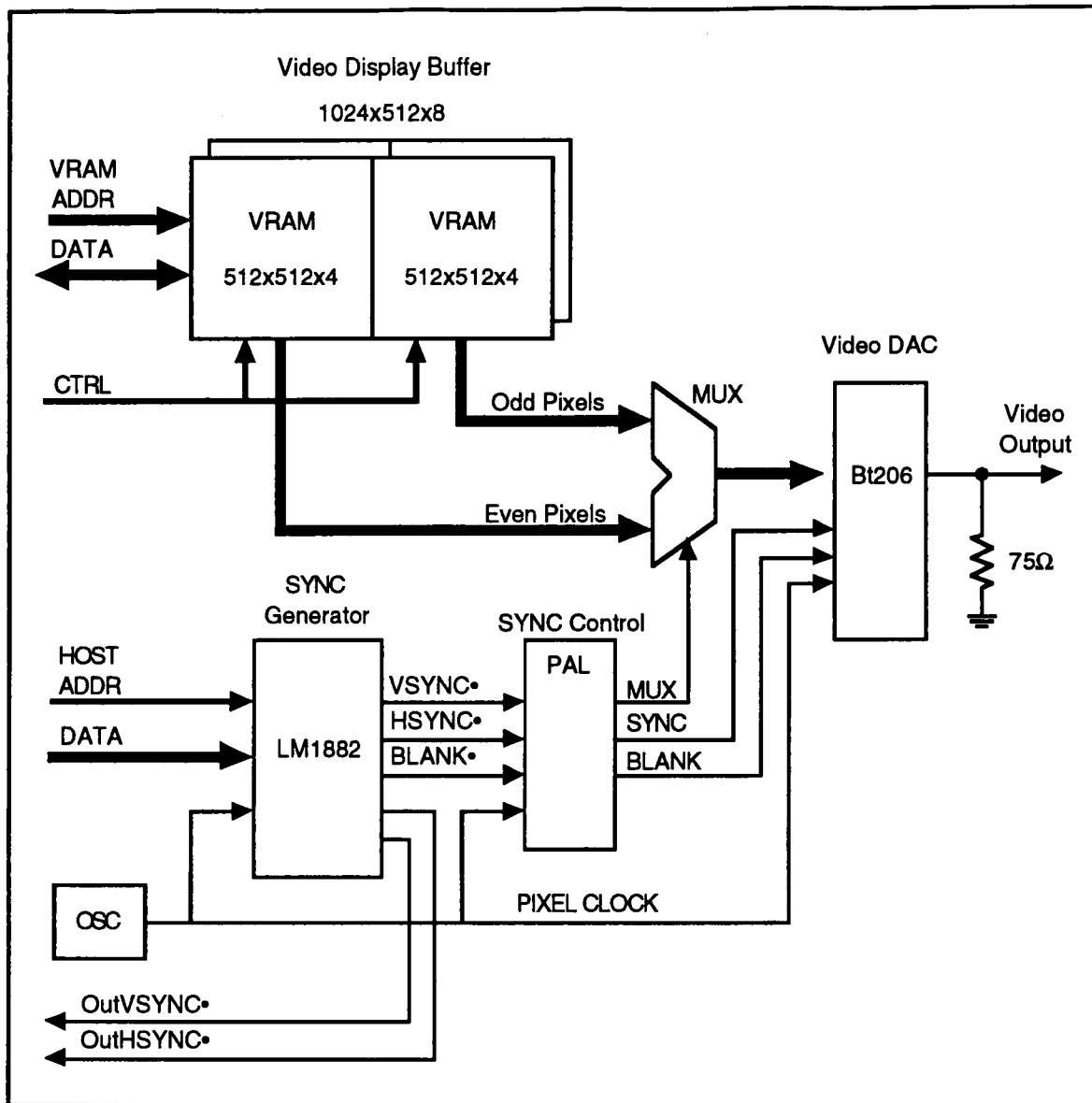


**Figure 6-6. Video Display Interface Block Diagram**

#### **6.4 INTERPOLATION FILTER**

The video interpolation filter circuit was designed to support several different interpolation filter kernel sizes. The filter is designed using a single high speed multiplier-accumulator (MAC). A functional block diagram of the interpolation filter circuit is shown in Figure 6-7. The MAC was implemented using the TRW LSI Corporation's TMC2208 which is an industry standard device. The MAC contains two 8-bit inputs and two separate control clocks. The ClkXY clock strobes in each new input set and performs the

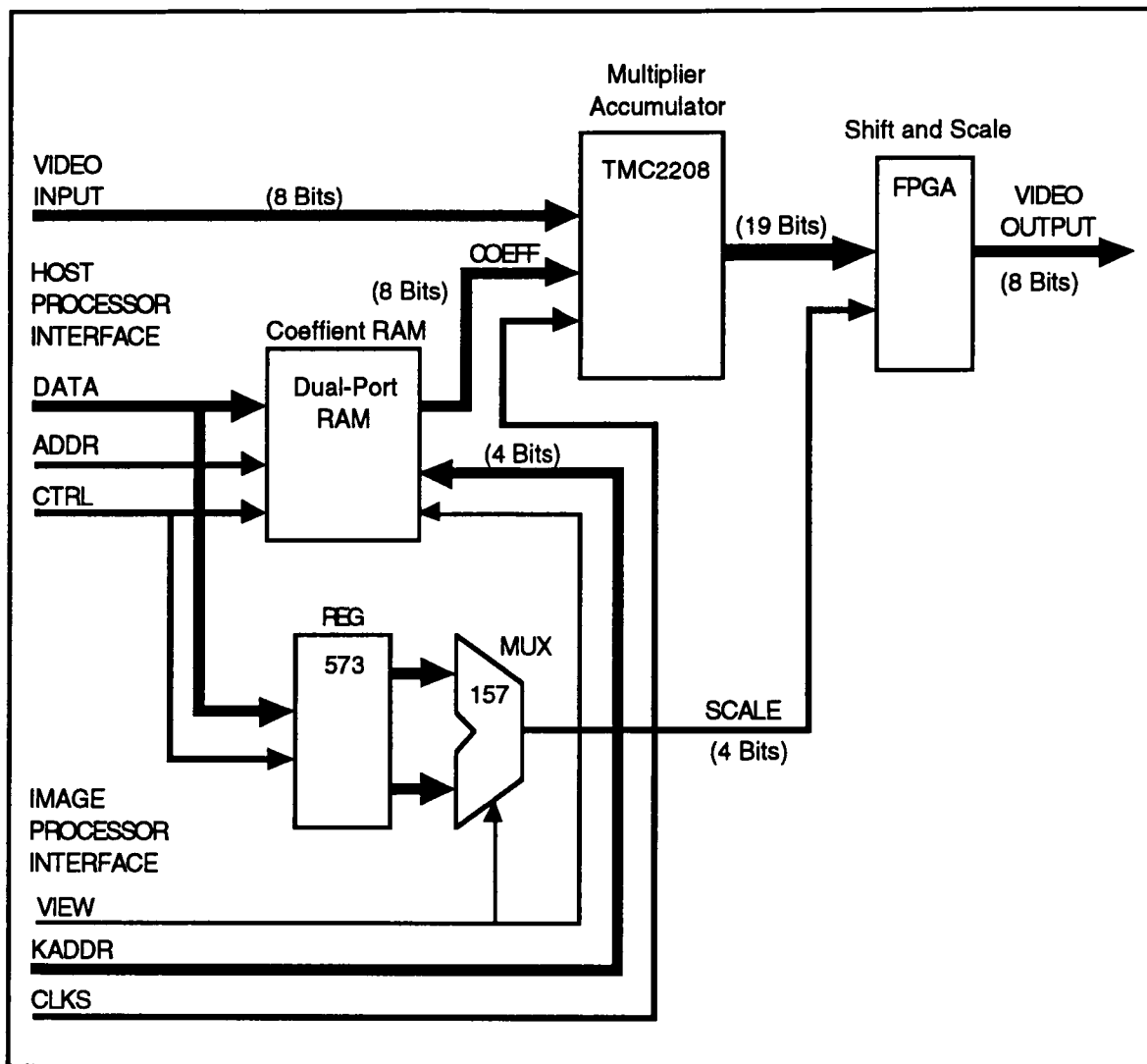


**Figure 6-7. Interpolation Filter Block Diagram**

multiplication. A separate ClkP signal generates the accumulated products for each sample. A third signal, ClkACC, controls the accumulation mode of the device. When ClkACC is asserted a new product is calculated without accumulation. This provides the capability to start a new interpolation kernel. When ClkACC is asserted, each new product is accumulated with the previous product. The TMC2208 maintains an internal 19-bit accumulator so that large interpolation kernels can be supported. The 19-bit output from the MAC is fed into a barrel-type shifter in order to scale the value to the correct 8-bit format. The MAC output barrel shifter was implemented using a single 44-pin Actel FPGA device.

The coefficients for the interpolation filter convolution kernel are generated using a single 8-bit dual-port RAM. The dual-port RAM implementation provides two main functions. First, two sets of interpolation kernel can be stored in the dual-port RAM. Each coefficient set can be selected by the View input. The other function of the dual-port RAM is to allow the host processor to update the next filter kernel coefficient set while one set is being used. The dual-port RAM implementation provides the ability to maintain different filter kernels for each active window.

## 6.5 SUMMARY

The prototype Omniview camera system design uses a single board for the video processing functions. The video board contains a 2048 by 2048 by 8-bit per pixel input video capture buffer and genlock system. The genlock video digitizer provides the capability to synchronize to RS-170 type video signals with negative composite synchronization signals. The composite sync information is detected by a genlock controller where it is split into the horizontal and vertical sync signals. The sync signals are then passed to a VRAM controller and arbitration block. The VRAM controller provides a multi-port interface and video memory synchronization signals to allow arbitration between the image processor, the host processor, the video transfer controller, and the refresh generator. The arbiter provides a prioritized arbitration scheme to allow access to the video memory.

A black-and-white output display provides a RS-170 standard video interface and is capable of driving a doubly terminated 75  $\Omega$  coaxial cable. The output display can be programmed to provide standard interlaced 60 field-per-second video information and can be programmed for other video standards such as non-interlaced and PAL.

An interpolator and filter block was designed to support both bilinear and cubic convolution kernels. The filter block provides the capability to implement different filter functions for each output display window.

The video memory controller and arbitrator was discussed in this chapter. An input and output video memory controller were designed using the Actel field programmable gate array devices. The FPGA designs operated at a system clock frequency of 20 MHz. Each video memory controller provides prioritized arbitration between the image processor, the host processor, the video synchronization and transfer control, and refreshes.

The video processor board was implemented on a 6-inch by 8 -inch printed circuit board. There are two connectors on the board which provide the system interfaces. One connector provides the host processor interface and control signals while the second connector provides the video memory control signals generated by the input and output video memory control logic.

## **CHAPTER 7**

### **OPERATING SYSTEM SOFTWARE**

The operating system and application programming support for the Omniview camera system were considered early in the design phase of the project. The goal for the Omniview application software and operating system was to provide the greatest degree of user flexibility while maintaining performance and minimizing software development time. The Omniview hardware and host microprocessor architecture were designed with the end user in mind. One of the goals of the Omniview system is to provide the ability for end users to be able to integrate the Omniview camera system into special purpose viewing applications. The user should be able to integrate high level software control and enhancements on top of the basic Omniview operating software and application routines. The C language was chosen for the operating system and application software development because of its fast compiled response and its ability to use libraries for software development.

#### **7.1 MONITOR PROGRAM DESCRIPTION**

The development of the Omniview host microprocessor operating system was enhanced significantly by Intel Corp. support. The operating system and monitor for the Omniview system were developed using the Intel NINDY monitor program supplied by Intel's embedded products development group. The NINDY monitor provides the user with most of the program support and debugging routines necessary for developing embedded hardware applications. The NINDY monitor C language source code was purchased along with an 80960SB development and evaluation system. The NINDY operating system was then ported to the Omniview host processor.

The NINDY monitor program was chosen for the Omniview host processor operating system because it provides the ability to perform hardware and software testing using a simple command-line interpreter. The monitor interface is designed to

communicate with most serial communication programs. The NINDY monitor uses a command line syntax that recognizes several useful commands for hardware interrogation, debugging, and testing. There are several forms of memory display and manipulation including display of byte, short, double word, triple word, and quad word data types. There are also commands for displaying floating-point data and operands. Display and modification of memory data is also supported. The monitor program also allows the user to display and modify the 80960 register contents interactively.

The NINDY monitor also includes facilities for debugging user application code. Single-step commands and instruction tracing support provide debugging support during program execution. The monitor program also supports up to two hardware breakpoints which can be inserted into the application program. A disassembler is included for examining program code in memory. The disassembler displays the instructions and data in memory in the 80960 assembler mnemonic format.

The NINDY monitor provides two types of object file downloading to support application development. The monitor provides two commands DL and DF that allow pre-compiled object files to be downloaded into either SRAM or FLASH. The download program uses and recognizes the XMODEM file transfer protocol and can be used to program the FLASH memory during the download operation.

The Macintosh personal computer was chosen as the host system for Omniview software development. The communication between the Macintosh host and the Omniview system is supported through the Red Ryder application software. Red Ryder is a complete communication program that contains support for uploading and downloading software programs using several different formats including the XMODEM transfer protocol and a text transfer function. Red Ryder also supports different terminal and user interface formats such as VT100 terminal emulation.

## **7.2 MEMORY SUPPORT CONSIDERATIONS**

The Omniview system uses both EPROM and FLASH memory technology in addition to SRAM memory in order to enhance the ability to develop and modify application programs for the end user. The EPROM, FLASH, and SRAM memory each occupy the same amount of host address space. One enhancement to the development process that has been used by the author successfully in many embedded applications is the

use of shadow memory and memory swap implementations to enhance the application development process. The Omniview system supports this development approach by providing both EPROM and FLASH memory technology. The NINDY object file uploading capability in conjunction with the memory swap feature significantly enhanced the application software development process.

The Omniview host processor uses EPROM to contain the main NINDY operating software and boot program while the FLASH memory provides storage of application programs that are user dependent. The Omniview system uses a front panel dip switch that indicates to the monitor program whether to execute an application program that resides in the FLASH memory or to continue executing the monitor program. After the host microprocessor has finished booting and initializing the communication hardware, the monitor program reads the front panel configuration switch and stores the value in a global variable. Each step of the boot process then tests the configuration switch settings in order to determine the operating conditions. The first position on the front panel switch is used to indicate whether a valid application program is programmed in the FLASH memory. If the switch is asserted, the monitor program passes control to the application program which continues the configuration process, otherwise the monitor program continues. Another position on the front panel configuration switch is used to swap the EPROM and FLASH memory spaces. The EPROM and FLASH swap feature provides the ability to test new EPROM operating software changes without having to erase and program the EPROM memory devices. The host processor operates identically whether running the FLASH memory program or the EPROM memory program since the wait state implementation for both are the same. This allows software debugging to be accomplished in a more efficient manner because a new operating system kernel can be downloaded into the FLASH memory and tested without having to erase and reprogram the EPROM devices.

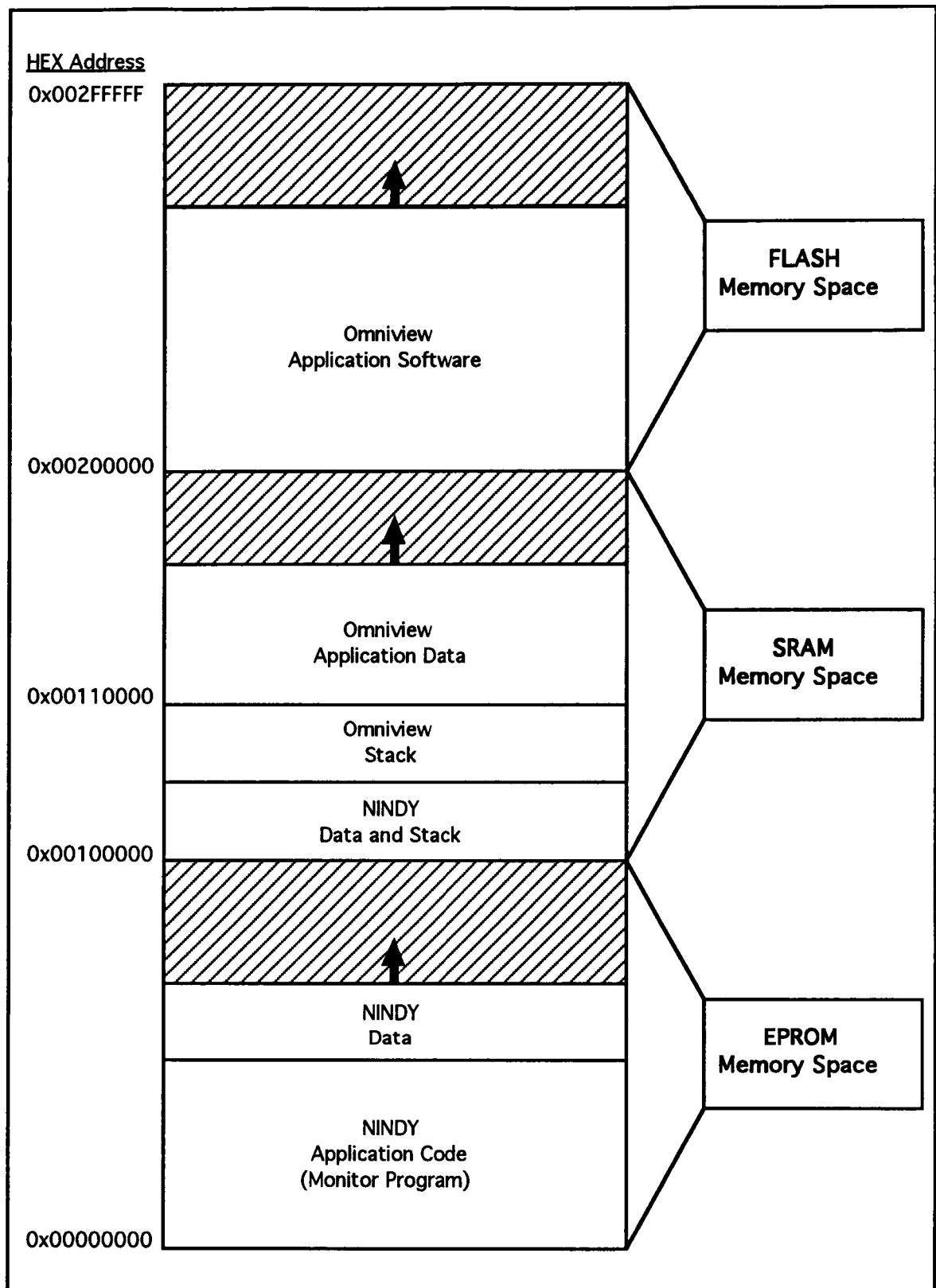


**Figure 7-1. Omniview System Software Memory MAP**

### **7.3 PROGRAMMING TOOLS**

The NINDY monitor and the Omniview application software are written in the C language and 80960 assembly code. The C language support provides the ability to support different users application software through high-level and low-level library routines. Almost all of the NINDY and Omniview application software are written in the C language.

#### **C Compiler and Assembler**

The application and program implementation for the Omniview host processor were developed using the Intel 80960 C compiler and assembler languages. The Intel C compiler (iC-960) is provided for DOS based personal computers. The iC-960 compiler provides cross-compilation and assembly code support for the entire 80960 microprocessor family and can be targeted for the particular architecture such as the 80960SB microprocessor which has the on-chip floating point execution unit. The iC-960 C compiler implementation conforms to the American National Standards Institute (ANSI) standard for the C language and is a two-pass optimizing compiler.

#### **Library Support**

The C compiler for the 80960 processor family comes with the basic low level and high level C software library support modules. The NINDY operating system is also provided with libraries that are linked in with the application software in order to use the low level operating system services. The low level NINDY libraries provide the basic read/write and I/O capabilities. In addition, an Omniview host specific library was created to handle host specific interface routines and startup code to support application code development.

### **7.4 APPLICATION SOFTWARE**

The Omniview application software is broken down into several C source code and assembly code files. A brief description of the main software application modules is given to provide a general overview. The Omniview camera system provides a unique implementation for remote viewing because all of the camera control functions are

programmable. The software can be manipulated to support end user requirements for many applications.

### **Virtual Camera View Support**

The Omniview prototype software implements four virtual cameras for viewing and display. The Omniview system can any or all of the virtual displays at any time. The prototype software offers the capability to display one, two, or four virtual cameras at any time. The image processor and host processor routines maintain the parameters updates for the active camera views. Each virtual camera view is implemented using a global structure that contains all of the high level and low level parameters that determine the attributes for the virtual camera. The pan, tilt, rotate, magnification, and location of the virtual camera can be updated at any time whether or not the view is actively displayed. A virtual camera can therefore be manipulated in the background to point in any direction before it is displayed on the monitor. This provides the ability to cycle between many different fixed viewing directions in a typical remote viewing application.

### **Transform Support**

The Omniview image transformation parameters are calculated within the file `xform.c` and contains three main routines. The first routine, `calculate_ptz_angles()`, is used to convert from the normal pan, tilt, rotate, and zoom angles, to the Omniview zenith, azimuth, and rotation angles. The routine provides simulation of a mechanical pan/tilt mounted camera. The second routine, `rotate()`, uses the angles determined in `calculate_ptz_angles()` to calculate all of the Omniview image processor transform parameters for the particular viewing angle. The transform routines operate on the virtual camera structures and provide the ability to manipulate any of the virtual camera viewing directions at any time. The effect is seen only when the virtual camera display is active on the display. The third main routine provides display parameter update to the image processor. The image processor is designed to operate using the last set of transform parameters that were programmed. The virtual camera display maintains its viewing direction until the display parameter update routine is called. When the image processor signals the end of the current transformation for the virtual camera, it interrupts the processor. The interrupt service routine then copies the newly updated parameters to the image processor parameter registers and saves a local set for later use. If no new parameter

updates are needed, the interrupt service routine uses the last set of parameters for the virtual camera.

## **Graphics and Text Support**

The Omniview graphics and text drawing routines are provided in the two files `omnidraw.c` and `text.c`. The main graphics drawing support is contained in `omnidraw.c`. The main routines for graphics drawing include line, point, rectangle, and block filling routines. The rectangle and block fill routines offer the capability to erase, paint, and invert either a boundary outline rectangle or a completely filled rectangle to either the input or output video buffers. The graphics drawing routines reference the global structure, `TheGraphPort`, to determine the drawing parameters, such as the graphics buffer size, the number of vertical and horizontal pixels, and the offset base address to the graphics buffer.

The file `text.c` provides two basic high level functions, `drawchar()` and `drawstring()`, that provide all of the graphics text drawing support. The `drawchar()` routine draws a single character at the current drawing point while the routine `drawstring()` draws a graphics text string at the current drawing point. The text routines reference a global graphics structure, `TheGraphPort`, that contains the current text size, the foreground and background colors, the font size, the current drawing point, and a pointer to the graphics buffer. The text and graphics drawing routines can draw to either the output display buffer or the input video capture buffer.

## **External Interface Control**

The external interface and control for the Omniview virtual cameras is provided by two main application files, `dsphand.c` and `dspctrl.c`. The `dspctrl.c` file contains the main control interface that communicates with an external computer or remote hand-held controller for controlling the Omniview virtual cameras. The remote communication interface uses a special protocol command structure in order to prevent system hang ups during operation. The protocol assigns the sign bit of the byte oriented communication for indication of commands. The data bytes are implemented as 7-bit numbers with the sign bit cleared. The control software determines when a possible command is being sent over the communication port through sign-bit detection and also compares the command to a list of valid commands that are implemented.

## 7.5 SUMMARY

A prototype Omniview system was implemented and the operating system and application program development was discussed. The Omniview system provides two features that enhance the operation and allow the Omniview system to be easily embedded in user applications. The host processor is an Intel 80960SB 32-bit RISC architecture microprocessor.

The Omniview system development process was enhanced significantly by the use of the NINDY monitor program available through Intel Corporation. The NINDY monitor C source and assembly code were purchased from Intel and ported to the Omniview host system. Some commands were added to the monitor program for Omniview specific support. The NINDY monitor program supports interactive debugging and object file uploading and provides many functions to support hardware programming and development.

The Omniview host processor design uses FLASH technology memory devices to improve and enhance the development of embedded applications. The FLASH memory provides the capability to erase and reprogram the Omniview application to meet users requirements. A set of pre-defined and pre-compiled libraries included with the Omniview system allows the user to focus on unique application issues that are unique to his system without having to develop new operating programs.

The startup and interrupt drivers and image processing support programs have already been developed and can be included with user application programs. This was considered necessary for the Omniview system because users have varied viewing situations and operating requirements. For instance, the Omniview system can be used in security and surveillance applications to simply monitor and survey different areas of a room or area and may need only fixed views with simple switch input control. A military user might use the Omniview system for remote flying and control of an Unmanned Aerial Vehicle (UAV). In this case, the need for stabilization and rotation of different views is important. The Omniview system can support each of these situations by uploading different application programs into the FLASH memory.

## **CHAPTER 8**

### **RESULTS**

Camera viewing systems are abundant in surveillance, inspection, security, and remote sensing applications. Remote viewing is critical for robotic manipulation tasks, and is often performed by cameras attached to mechanisms that provide the pan, tilt, zoom, and focus capabilities. Close viewing is necessary for detailed manipulation tasks while wide-angle viewing aids positioning of the robotic system to avoid collisions with the work space.

In order to minimize the size of the camera and orientation mechanism, a camera system was developed for remote viewing applications that utilizes fisheye optics and electronics processing to provide pan, tilt, zoom, and rotational capabilities within a hemispherical field-of-view with no moving parts. The Omniview camera approach is based on the properties of a fisheye lens which provides a complete hemispherical field-of-view image. A high speed image transformation processor has been developed that transforms portions of the image into a corrected image perspective for display on a display monitor.

The Omniview camera system has several advantages over conventional pan-tilt-zoom camera systems. The Omniview camera implementation allows multiple images to be simultaneously displayed from a single omnidirectional camera to provide from one location. The transformation is accomplished electronically providing complete programmable control over viewing parameters. Image sizes, viewing directions, scale and offset, etc. may be adjusted to fit operator needs. Since the fisheye image is symmetrical the camera azimuth can be adjusted electronically to compensate for the camera orientation.

#### **8.1 SOFTWARE OMNIVIEW DEMONSTRATION**

The initial demonstration of the feasibility of the Omniview transformation approach was performed using the Macintosh personal computer. A demonstration version of the

Omnview approach was developed using the C language on the Macintosh. The application was written using the Lightspeed C programming environment and made use of the graphical user interface support provided by the Macintosh. A menu driven program was written that would allow the user to select a previously stored fisheye image. The images used in the demonstration were obtained using photographs taking with a standard 35-mm Nikon camera equipped with an 8-mm Nikon 180 degree fisheye lens. The photographs were then scanned into the Macintosh PICT image format using a gray-scale 300 dots-per-inch flat bed scanner. The photographs were scanned at a resolution of about 1000 pixels across the diameter to simulate the resolution of a high resolution 1024 by 1024 pixel CCD camera using a fisheye lens.

A photograph of the software demonstration showing a typical fisheye image and a corrected image is shown in Figure 8-1. The software demonstration program provides a menu driven interface to allow the transformation of the input image into corrected output images. The software provides the ability to select the image center and the radius of the fisheye image by using a mouse input and cursor. Once the parameters of the input image have been selected, the user simply executes the "Select View" command under the menu item list. The software then waits for the user to click the mouse in the input image window on the point where the transformation is to be applied. The software then performs the fisheye correction using the particular image point as the center. For the demonstration system, the output image size was fixed at 320 horizontal by 240 vertical pixels to simulate the image resolution of standard low resolution NTSC display monitors.

Another command from the menu item command list provides the capability to manually adjust all of the important image parameters for the transformation. A photograph of the Macintosh display showing the parameter input window is shown in Figure 8-2. The parameter list input provides the user with a means of logging the current image center and radius measurements and to manually adjust the viewing angles.

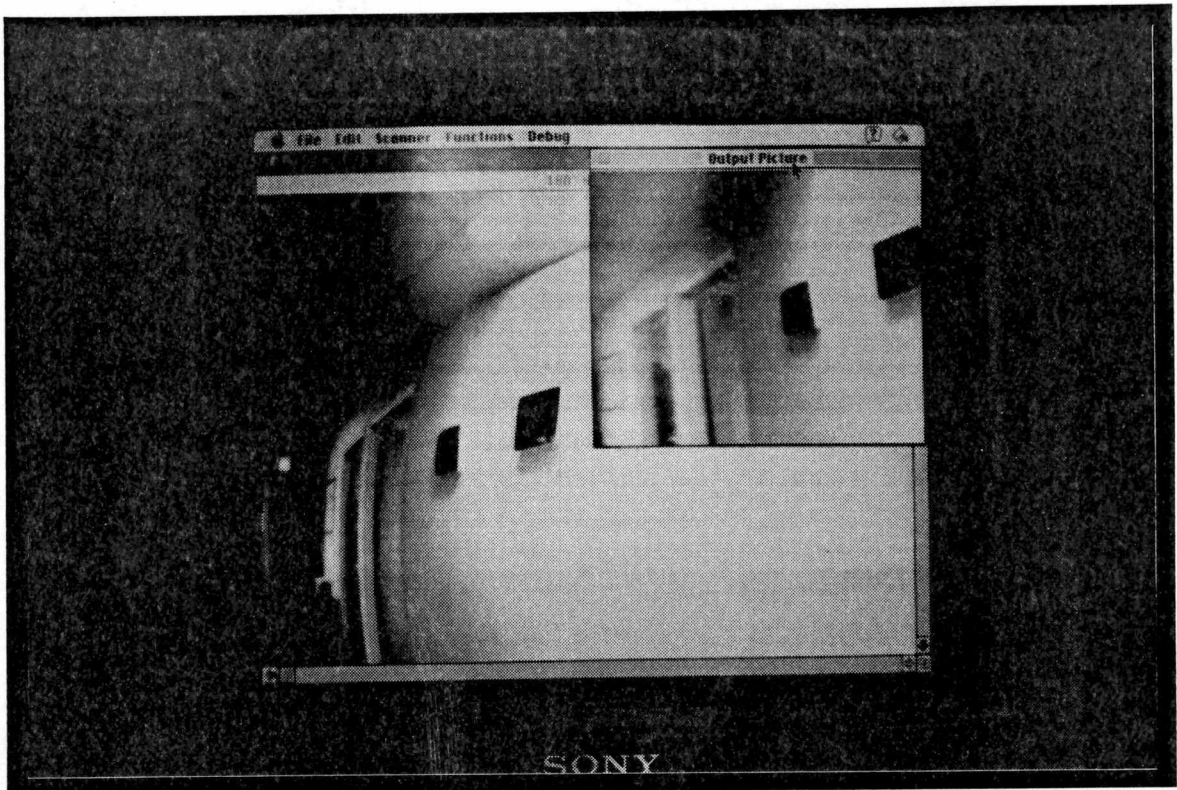


Figure 8-1. Photograph of Software Demonstration

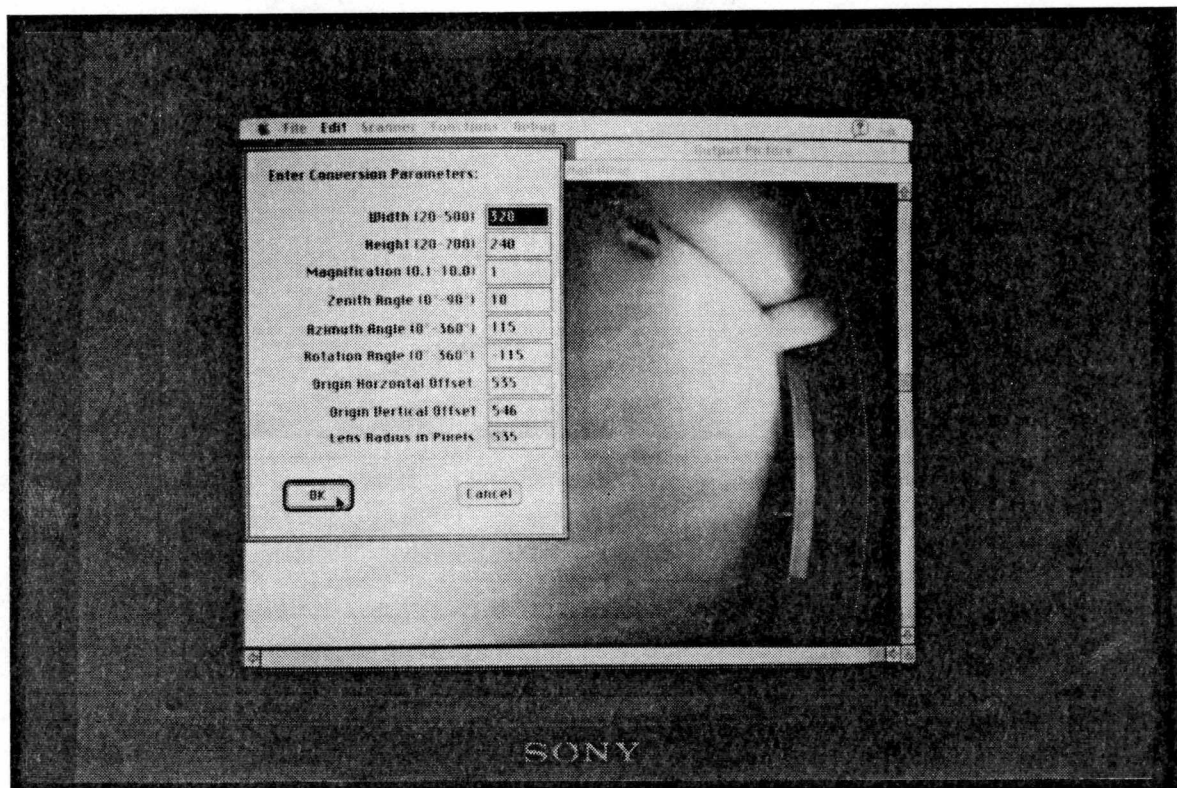


Figure 8-2. Photograph of Software Demonstration Parameter Input

## 8.2 WIRE-WRAP DEVELOPMENT SYSTEM

The initial hardware demonstration of the Omniview camera system was accomplished using a single wire-wrapped prototype development board. The wire-wrap prototype board was assembled in order to test some of the hardware approaches to the Omniview system for feasibility. A photograph of the wire-wrap development board is shown in Figure 8-3. The wire-wrap development board proved to be extremely useful for testing the design approach for the Omniview image processor.

The wire-wrap demonstration system used the 16-bit 80C196 microcontroller as the host processor. The 80C196 microcontroller embedded core has been use successfully by the author for numerous projects. The operating system for the wire-wrap prototype system was written entirely in FORTH. An important effort in the development of the wire-wrap prototype was the implementation of a 96-bit precision arithmetic software package to support the programming of the image processor transformation coefficients. The arithmetic package included a 48-bit multiply function that produced a 96-bit result. A high precision division routine was implemented that allowed a 96-bit dividend to be divided by a 48-bit divisor to yield a 96-bit quotient. Other high precision operations included a 48-bit SIN and COS look-up table for integer degrees, and a 48-bit precision square-root function. The FORTH system proved to be invaluable in the development and demonstration of the Omniview system.

A photograph of the camera and lens setup that was used in the wire-wrap demonstration system is shown in Figure 8-4. The wire-wrap demonstration system used a Videk Megaplug camera for the image input. The Megaplug camera is a 1.4 million pixel CCD camera which uses a Kodak high resolution CCD image sensor as the sensing device. The Kodak sensor is a full frame CCD image sensor arranged in a 1035 vertical line by 1320 horizontal pixel format with 6  $\mu\text{m}$  square pixels. The size of the CCD image sensor array is approximately 6.2 mm by 7.9 mm. The CCD sensor was designed for use with 2/3 inch format C-mount cameras that are prevalent in the CCTV industry.

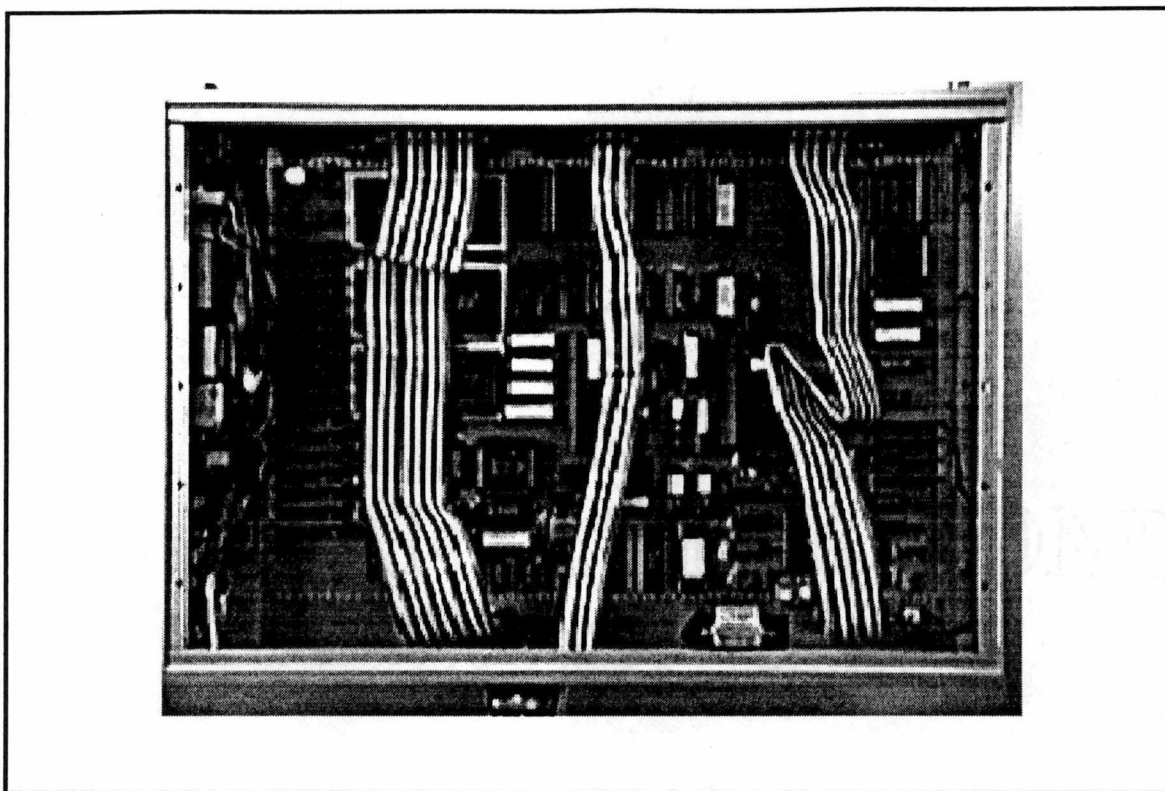


**Figure 8-3. Photograph of Wire-Wrap Development Electronics**

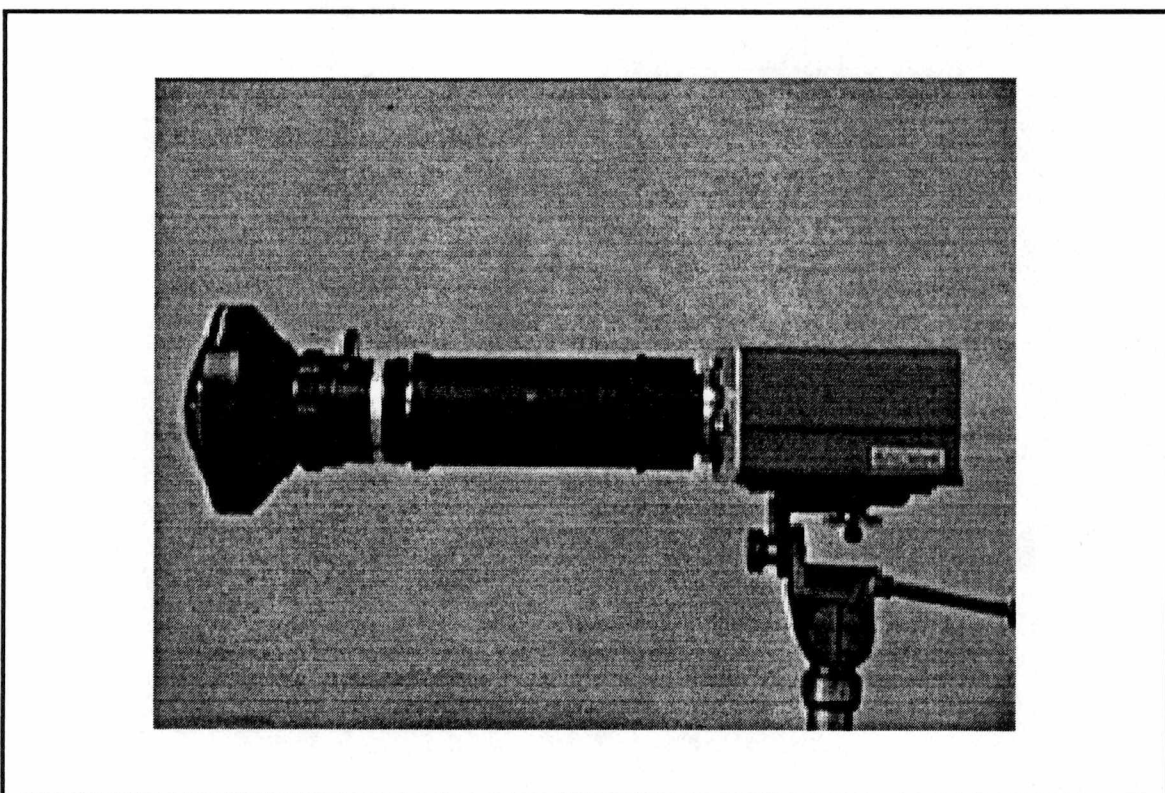


**Figure 8-4. Photograph of Demonstration Camera System**

Several different approaches were taken to implement the fisheye optics for the Videk camera. The simplest approach was accomplished by using a Nikon 8-mm F-mount lens which produces a 180 degree field-of-view when used with the regular Nikon 35-mm F-mount cameras. Since the Nikon fisheye lens produces a 23 millimeter circular image on the 35-mm format at a back focal distance of 43.5 mm, a lens reduction element was necessary in order to adjust the image size to match the Videk camera CCD image sensor array format. Several different approaches were attempted and a final solution was found that was satisfactory for the demonstration system. A Nikon lens reduction element was located that provided an overall image reduction of about 3.8 times or about 26 percent of the original size. The image reduction element was over 6 inches long and contained several lens elements so that the final image dynamic range and contrast were severely limited. The wire-wrap development effort, however, led to a successful demonstration of the Omniview camera system approach.

### **8.3 OMNIVIEW PROTOTYPE SYSTEM**

A photograph of the prototype Omniview camera system is shown in Figure 8-5. The Omniview prototype electronics includes five connectors which provide system interfaces. The computer control and remote control communication interfaces are provided by two DB-9 connectors mounted on the front panel. The communication interfaces are implemented as RS-422 type transmitter/receivers for long distance communication capability. A remote hand-held controller provides the ability to remotely control the Omniview system and connects to the left communication port. A remote computer interface port is included for automated and computerized operator interface control implementation and connects to either the left or the right communication port. The Omniview system uses an external 120V ac input and +5V dc output power supply that connects to a DB-9 type connector on the rear of the Omniview system. A DB-15 type connector on the front panel provides the genlock camera interface which can be used to digitize of several types of cameras. External programmable digital horizontal and vertical syncs are brought out through the external DB-15 video interface connector so that users may generate standard or non-standard video syncs for different camera digitizing requirements. The output display is brought out through a 75 ohm BNC type connector on the front panel. The output display interfaces to most RS-170 type black-and-white monitors and provides NTSC standard black-and-white interlaced video or PAL type video

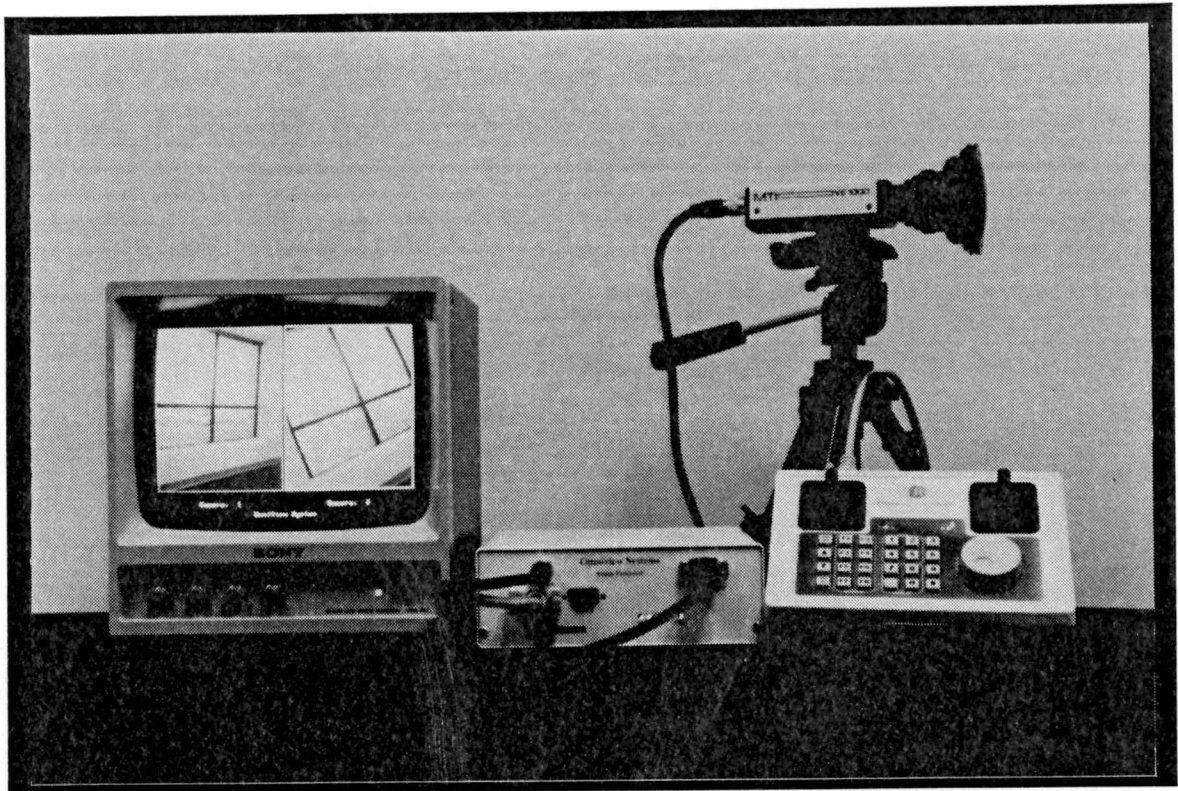


**Figure 8-5. Photograph of Prototype Omniview System**

information. Other formats such as non-interlaced video can also be supported by the output display interface.

The Omniview prototype electronics system is shown in Figure 8-5 and the electronics boards are shown in Figures 8-6 through 8-9. The Omniview camera system consists of three main processing boards - the image processor, the host processor and image buffer interface arbitration logic, and the video processor board. The electronics boards are 6 inch by 8 inch in size and are housed in a 6 inch by 8 inch by 2.5 inch electronic enclosure. The three electronics processing boards mount together via hardware standoffs. Connectors on each board interface provide signal and power connections and bus interfacing.

A photograph of the image processor board is shown in Figure 8-5. The image processor board contains all of the components that relate to the image processor design implementation and is a self-contained processor. The image processor board contains the complete image processor core design and functions as a self-contained unit. The image

processor board may be integrated into other image manipulation systems through the two 56-pin connector interfaces.

A photograph of the host processor board is shown in Figure 8-6. The host processor board includes the 80960 based host microprocessor core design and also contains the input video memory buffer and output video memory display buffer arbitration and interface logic. The host processor is the middle board of the Omniview electronics system and acts as the buffer and control interface between the host processor, the image processor board, and the video processor board.

A photograph of the video processor board is shown in Figure 8-7. The video processor board contains the input video camera genlock and gray-scale video digitizer interface, the input video capture buffer with up to 2048 by 2048 pixel video frame buffer, the filter and interpolation logic, and the output video memory buffer and output video display interface.

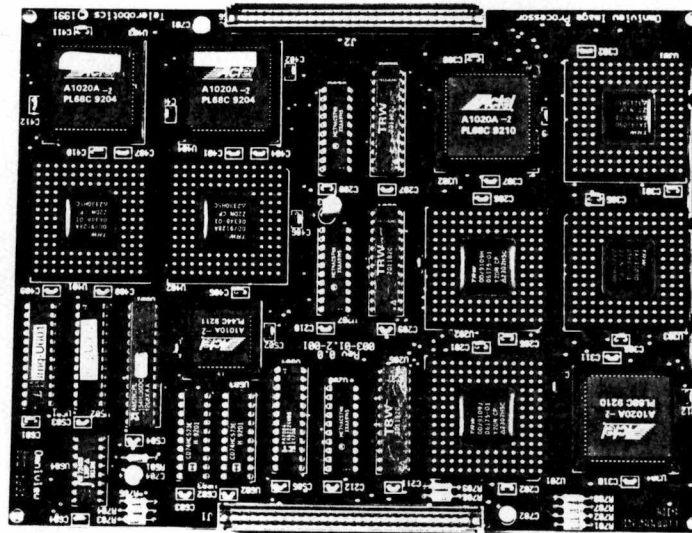


Figure 8-6. Photograph of Image Processor Electronics Board

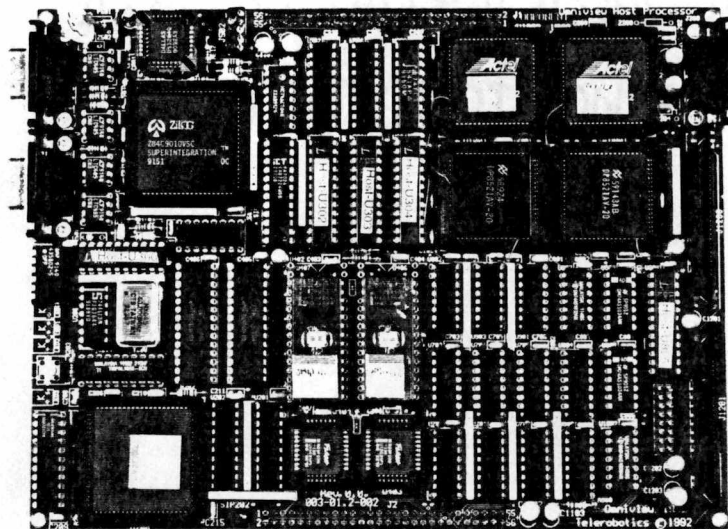


Figure 8-7. Photograph of Host Processor Electronics Board

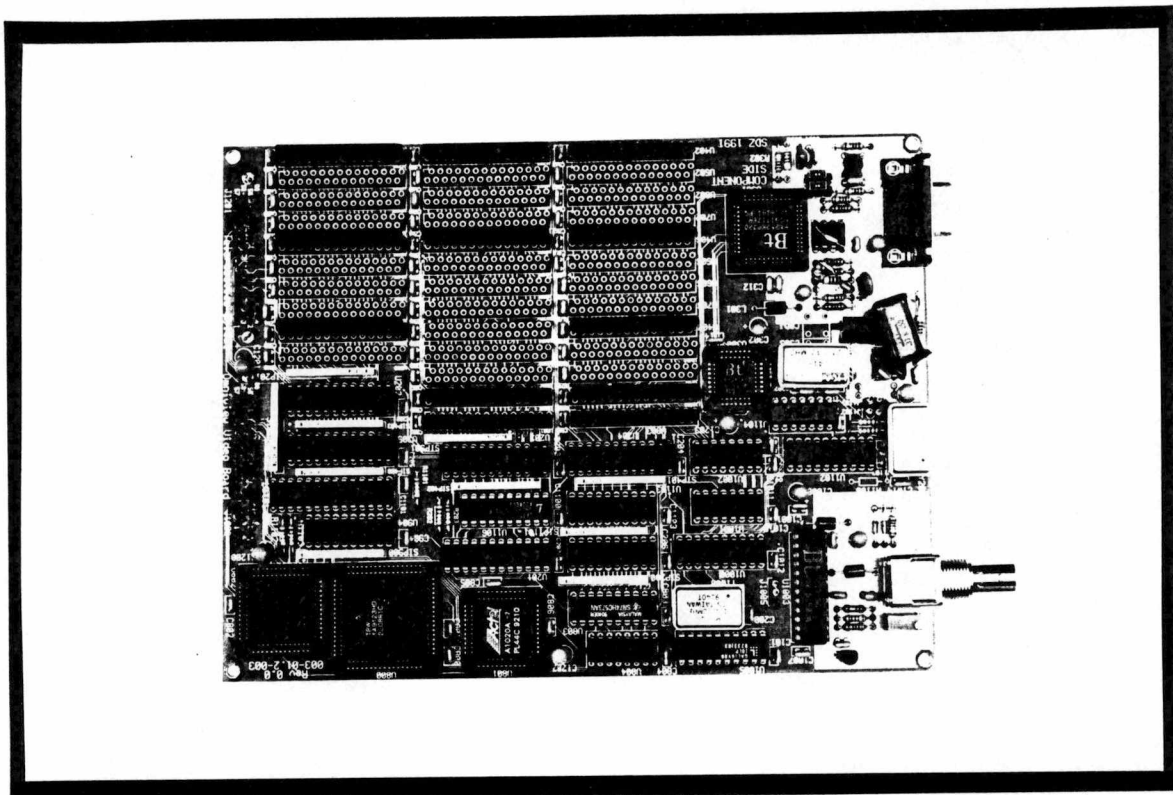


**Figure 8-8. Photograph of Video Processor Electronics Board**

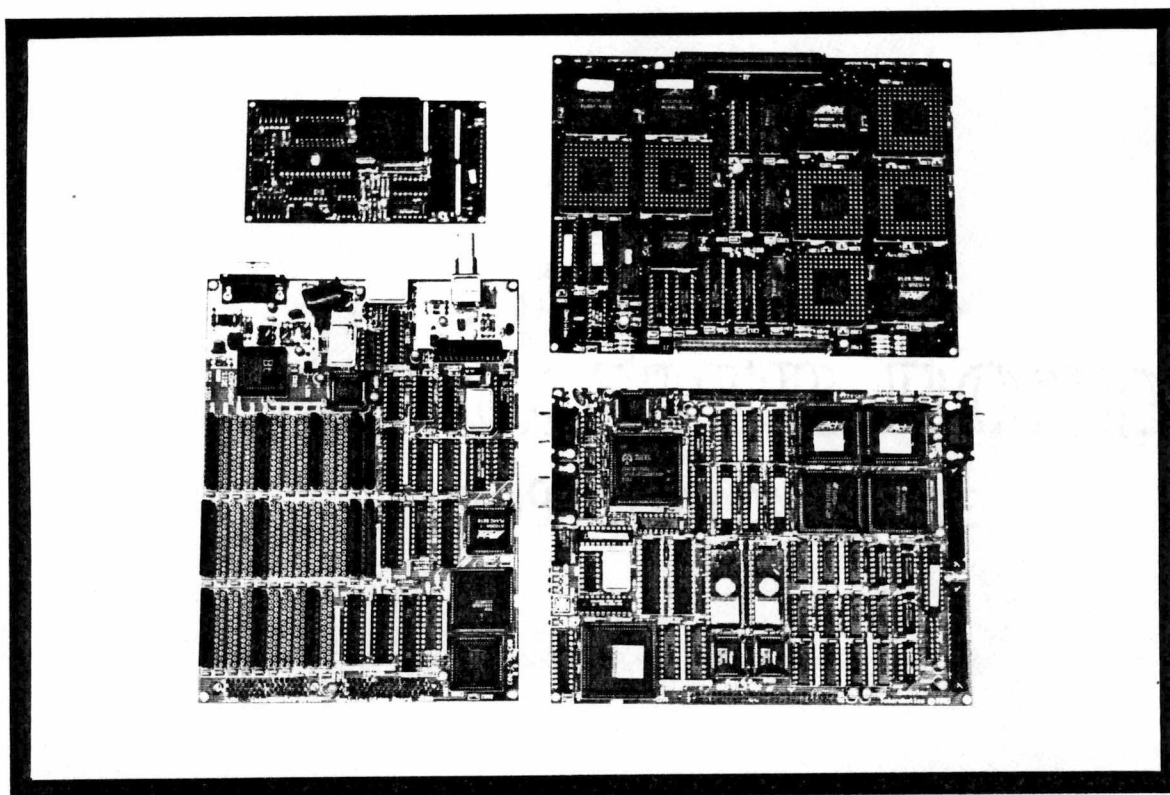


**Figure 8-9. Photograph of Omniview Electronics**

## 8.4 OMNIVIEW DISPLAY CONTROL

The Omniview display control operates similarly to conventional pan-and-tilt camera systems used in industry. The Omniview system provides unique image control capabilities over other systems. The ability to provide viewing over a full 180 degree fisheye image and the capability to select multiple images enhances the performance above conventional remote viewing systems. The Omniview approach can be used with various fields-of-view from narrow angle to full hemispherical images. A photograph of the Omniview system showing a wide-angle uncorrected image was shown in Figure 2-1. The picture shows the camera output with a 170 degree field-of-view.

Several different viewing situations are shown in Figures 8-10 through 8-12. The single display mode is shown in the photograph in Figure 8-10. The image being displayed uses a 480 pixel by 360 pixel window to display the transformed image. The display also shows the graphics display capability of the host processor. The window contains an outline bar to designate the viewing window being updated by the image transform processor. In addition, the bottom portion of the display contains some text graphics to display the current viewing parameters and the current virtual camera image being displayed. Some of the parameters being displayed are the rotation angle, the camera azimuth angle, the pan-and-tilt angles, and the magnification.

The dual display mode of operation is shown in Figure 8-11. The left and right images shown in the photograph demonstrate the capability of the Omniview system to provide independent virtual views. Each view can display independent rotation, pan-and-tilt, and magnification. The left image as shown in Figure 8-11 shows the operator in view while the second view shows the operator in view with the image rotated slightly and with a different magnification. A quad mode display is shown in the photograph in Figure 8-12 where four images are being displayed at the same time. The four views show different viewing directions and different magnifications. Each image in the quad display can be updated independently with the Omniview display controller interface. The Omniview display implementation is not limited to the quad display however. The Omniview system can display and transform as many images as the host processor update bandwidth will allow. The image processor pipeline processing sets a lower bound on the smallest image that can be displayed while the host processor interrupt processing. Currently it appears that this could be as many as 64 virtual images with an image size of 64 by 64 pixels.

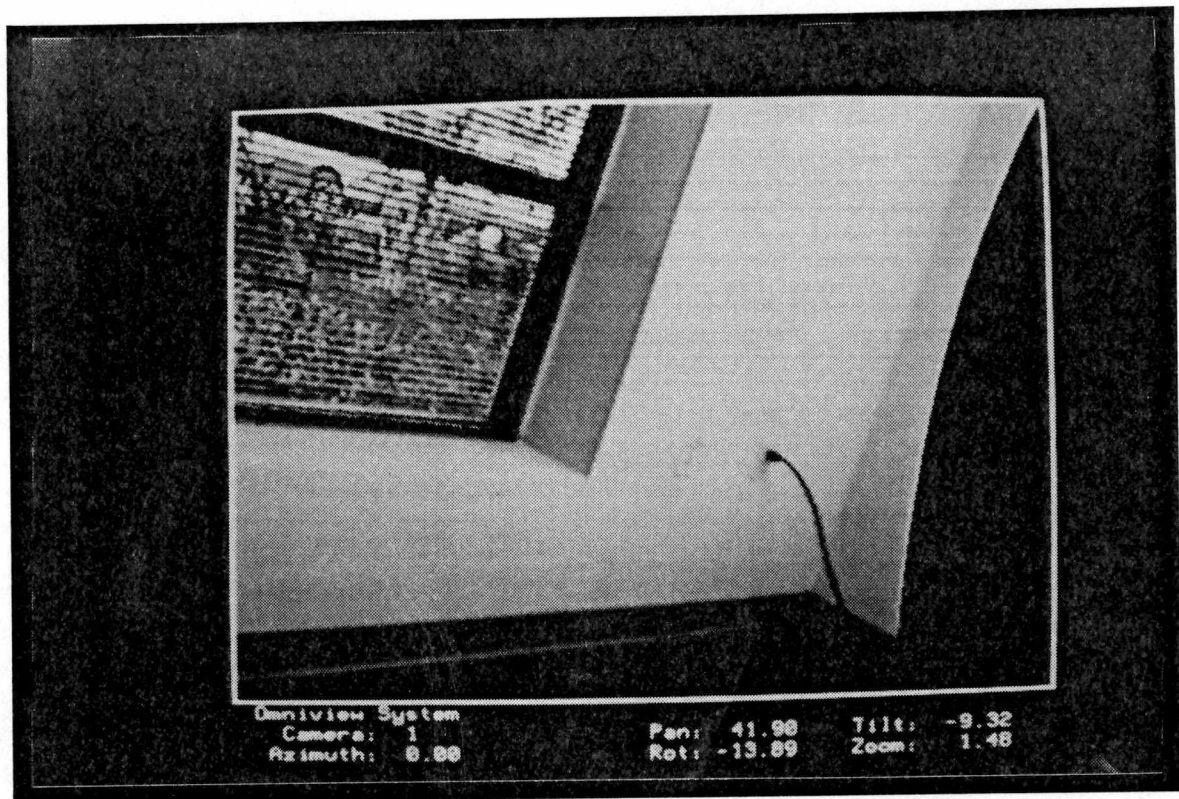


**Figure 8-10. Photograph of Single View Display**

## **8.5 FUTURE DEVELOPMENT**

The Omniview prototype camera system was developed to prove the technology and demonstrate a working system for potential customers. The present efforts involve significant design enhancements to the prototype system.

A higher frame rate version is current being developed to increase the transformation frame rate to 30 frames-per-second. The frame rate in the first version of the design was limited to about 22 frames-per-second for a 480 by 360 pixel image. The video buffer interface control logic and the video memory access speeds proved to be the limitation in the prototype system. The minimum cycle time per access was limited to about 250 nS giving a pixel rate of 4 MHz. The design goal for a complete 640 by 480 pixel output display is a 9.2 MHz pixel transform rate which corresponds to a 110 nS access cycle time. Current video memory technology can support address pipelined non-page-mode video memory accesses at a minimum cycle time of about 150 nS with access speeds approaching 135 nS in the near future.

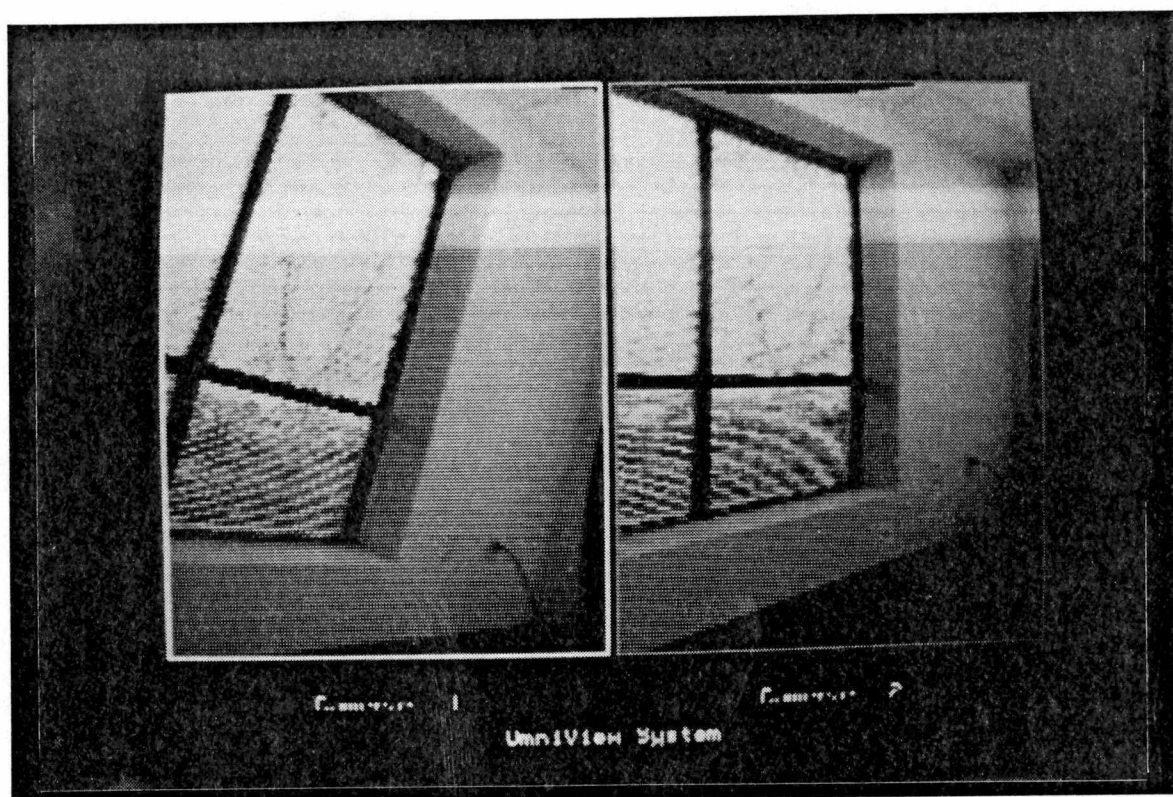


Figure 8-11. Photograph of Dual View Display

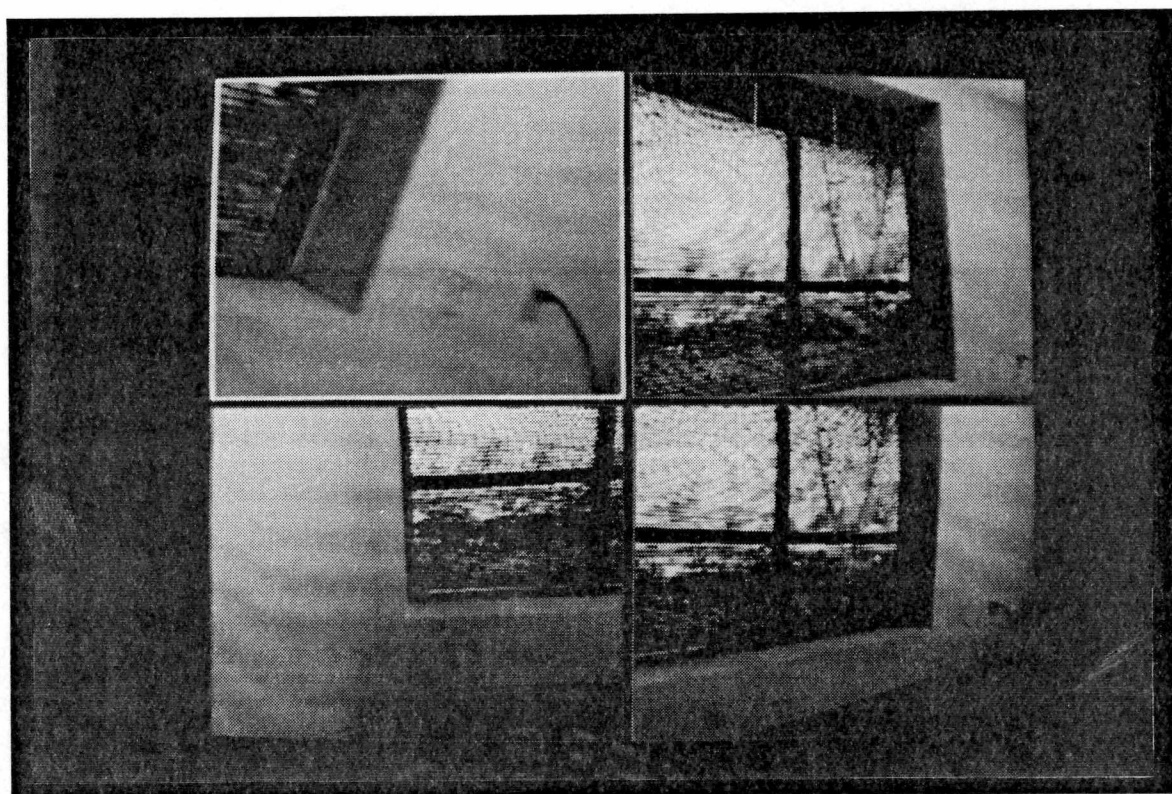


Figure 8-12. Photograph of Quad View Display

Current design enhancements involve designing a higher throughput video memory interface and arbitration controller that can support the 110 nS cycle times using a combination of address pipelining and page-mode interleaving. Other improvements to the current Omniview system involve the addition of color and black-and-white systems with higher quality genlock digitizer interfaces and improved filtering. A full 24-bit RGB color version that supports 1024 line high quality color and gray-scale cameras is being developed. The new design will incorporate color filtering and display systems and will also provide 30 frame-per-second transformation rates. The black-and-white systems will support up to 2048 by 2048 pixel cameras with frame rates up to 15 Hz.

## **8.6 SUMMARY**

The design and implementation of the Omniview camera system prototype was developed and presented. The Omniview system uses a patented transformation approach to provide electronic pan, tilt, rotation, and zoom over a complete hemispherical field-of-view. A prototype system was developed and successfully demonstrated to NASA Langley Research Center who sponsored this effort.

The Omniview implementation consists of three main electronics image processing boards. An image processor provides hardware speed real-time transformations and is dedicated to the execution of the Omniview transformation algorithms. A host processor board provides the operating system environment and the interface to remote control systems. A video board provides the input video digitizing and output video display and filtering. The entire system is contained in a 6 by 8 inch by 2.5 inch electronic enclosure.

The Omniview camera system provides a unique solution for remote viewing requirements. The full hemispherical field-of-view capability provides a wider coverage than most mechanical pan-and-tilt mounted camera systems. The electronic implementation of the pan, tilt, rotation, and zoom functions provides a very flexible system to meet various viewing needs. Slewing from one direction-of-view to another direction-of-view can be accomplished at full frame rates.

Potential applications of the Omniview system include remote viewing in constrained environments, inspection, object tracking, and surveillance applications. Omniview is ideal for multiple target acquisition and image stabilization applications due to

its multiple image handling and fast response capabilities. It can replace several camera systems in a remote viewing application.

## **BIBLIOGRAPHY**

# BIBLIOGRAPHY

- Actel Corporation, ACT<sup>TM</sup> Family Field Programmable Gate Array Databook, April 1992.
- Advanced Micro Devices Inc., PAL Device Data Book - Bipolar and CMOS, 1990.
- Brooktree Corporation, Product Data Book, Third Edition, 1991.
- Dallas Semiconductor Corporation, Product Data Book, 1990-1991.
- Eldon, John, "Non-Linear Operations with the TMC2301 Image Resampling Sequencer," Application Note TP-40, TRW LSI Products Inc., 1988.
- Eldon, Dr. John, "Performing Keystone Correction With the TMC2301," Application Note, TRW LSI Products Inc., July, 1989.
- Eldon, John, "Using the TMC2301 Image Resampling Sequencer," Application Note TP-37, TRW LSI Products Inc., 1988.
- Grass Valley Group, NTSC Studio Timing: Principles and Applications, Produced and Published by Grass Valley Group, Inc., 1988, Grass Valley, CA.
- Gamal, El, et al., "An Architecture for Electrically Configurable Gate Arrays," IEEE Journal of Solid-State Circuits, VOL. 24, No. 2, April 1989.
- Intel Corporation, NINDY User Manual, Intel Corporation, First Printing, April, 1990.
- Intel Corporation, i960<sup>TM</sup> SA/SB Microprocessor Reference Manual, Intel Order No. 270929-003, 1991.
- Kaplan, Wilfred, Advanced Calculus, Addison-Wesley Publishing Company, Inc., Reading Massachusetts, 1952.
- Moore, Will R. and Luk, Wayne, FPGAs, Abington EE&CS Books, Abington, England, 1991.

Signetics Corporation, "Modeling the PLL," Application Note, Linear Products Databook, Signetics Corporation, 1988.

Signetics Corporation, "An Overview of the Phase-Locked Loop (PLL)," Application Note, Linear Products Databook, Signetics Corporation, 1988.

TRW LSI Products Inc., DATABOOK - Data Converters and DSP Products, 1991.

Volder, J., "The CORDIC Trigonometric Computing Technique," IRE Transactions on Electronic Computers, Vol. EC-8, September 1959, pp. 330-334.

Williams, Fred, "The CORDIC Algorithm - Cast in Silicon," Electronic Engineering Magazine, September 1989, pp. 47-49.

Zimmermann, Steven D., Optimizing the Camera and Positioning System for TeleRobotics Worksite Viewing, NASA Contract No. NAS1-18627, Final Report, August 1988.

Zimmermann, Steven D., "A Remote Video Pan/Tilt/Magnify/Rotate Camera System with No Moving Parts," SPIE International Symposium on Optical Applied Science and Engineering, San Diego Convention Center, San Diego, CA, July 1992.

## VITA

Steven Dwain Zimmermann was born in Hohenwald, Tennessee on June 2, 1958. Attending the Lewis County public school system, he graduated Lewis County High School in 1976 where he was valedictorian. At L.C.H.S. Steve participated in academic, sports, and music activities. After graduation from L.C.H.S. he attended Tennessee Technological University at Cookeville, Tennessee in the fall of 1976 only to find that the lack of financial means or assistance prevented him from pursuing a degree there. After working for four years as a general contractor with his father as partner he then entered the University of Tennessee College of Engineering in 1980 specializing in analog electronic design and graduated in 1984 with highest honors and as top graduate in the college of engineering. While at U.T. he worked as a summer engineering student at Tennessee Technology for Energy Corporation in Knoxville, Tennessee and at Texas Instruments in Dallas, Texas. After attaining his Bachelor of Science in Electrical Engineering degree he began work at Oak Ridge National Laboratory (ORNL) where he was involved in specialized electronics and controls for robotics and robotics applications. He left ORNL and joined colleagues at TeleRobotics International, Inc. where he began pursuing a Master of Science degree in Electrical Engineering. At TeleRobotics, Steve designed and developed key products for the company and invented the patented Omniview camera system. He is currently working on improvements to the Omniview camera system design and applications of the Omniview technology. He shares his efforts with his wife Sherry and daughters Stephanie and Sarah.