DESIGN OF A REAL-TIME FLEXIBLE IMAGE PROCESSING SYSTEM

by

Anantha P. Chandrakasan

Memorandum No. UCB/ERL M91/27

18 April 1991

DESIGN OF A REAL-TIME FLEXIBLE IMAGE PROCESSING SYSTEM

by

Anantha P. Chandrakasan

Memorandum No. UCB/ERL M91/27

18 April 1991

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Acknowledgements

I would like to thank my advisor Professor Bob Brodersen for all the advice and support he has given to me over the last few years. He has been a great source of inspiration.

I would also like to thank Bill Baringer who has been a great supervisor to me during this project. He has helped tremendously in all phases of the project (including proof reading my report) and I especially thankful for his extreme patience.

I would like Professor Jan Rabaey for reading my report and making suggestions.

Special thanks to Jane Sun for her help in the design of the the VME interface and THOR simulations.

I would like to thank Phil Schrupp and Sue Mellers for their support for board design.

I would like to thanks all my friends who have helped me through my project.

Finally, I would like to thank my parents for all the encouragement they have given me during my study. I would also like to thank Mrs. Coe and my brother who have been extremely patient tolerating my constant complaints.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

Digital image processing refers to the processing of two-dimensional data. Image data is obtained by sampling and quantizing a raster-scanned analog representation of the image. The quantized picture-elements (also called pixels) are used as sampled data for image processing functions. A typical image processing system may use a frame size of 512 x 512 pixels and have an update rate of 30 frames/sec. Image processing has a wide range of applications including image enhancement, restoration, feature extraction, object recognition, and image compression/decompression[1][2][3].

Image enhancement brings out certain features of the image for future analysis or display. Examples include edge and contrast enhancement , and noise filtering. Image enhancement does not increase the inherent information in the image, but rather it emphasizes certain characteristics.

Image restoration removes known degradations from images. This may include the de-blurring of images degraded by noise introduced from the camera. Image restoration is concerned with filtering the image to minimize the effects of such degradations. Hence, given a noisy image, the image restoration problem is to find an estimate of the input. A "weiner" filtering operation represents a typical example of image restoration. Image enhancement differs from image restoration in that the former emphasizes more on the restoration of degraded images rather than extraction of image features.

Another major image processing application is feature extraction and recognition. Computer vision systems typically use these functions. The idea is to extract important

details from the image so that a computer can analyze and recognize the image. A typical application is in an assembly line where a vision system may detect and isolate faulty parts based on features stored in the computer.

Image data compression involves finding ways to minimize the "amount of data" required to represent an image or series of image frames while maintaining "acceptable" levels of visual fidelity. Image compression finds applications largely in storage and transmission. To transmit images defined over a 512 x 512 lattice, assuming a resolution of 24-bits per pixel and an update rate of 30 frames/sec, requires a transmission rate of 188 Mbits/sec. However, the maximum transmission rate is fixed by the bandwidth of the channel being used for communication. A compression of 125:1 is required to transmit the video sequence over a "T1" channel, at 1.5 Mbits/sec.

A first generation image processing system was designed to implement many of the above functions in real-time. NMOS custom ASICs were designed to perform real-time image enhancement, restoration, and extraction. These custom chips were interconnected to provide a real-time image processing system. The image processing functions were split up into two separate wire-wrapped boards. One board performs various real-time image filtering functions and image recognition[5][6] while the other performs real-time histogram and histogram equalization[7][8]. The system is based around a 21-slot VME card-cage. A Commercial A/D-D/A board and frame buffer-boards reside in the card-cage providing the required video interface and storage. A wire-wrapped multiplexor board[9], also residing in the card-cage, provides programmable routing of video data. The two image-processing boards reside outside of the card-cage and communicate with the interface and storage boards through video ribbon cables and connectors. Due to the limited routing capability of the multiplexor board, only one of the image processors (either the image processing/extraction board or the histogram board) can access the video setup.

The goal of this project is to integrate many of the existing video modules in the first generation image processing system into a flexible, compact, and robust system. This report describes the redesign of the existing image processing hardware, integrating and extending existing hardware for more robustness and flexibility. This system realizes a wide range of real-time image processing algorithms including image recognition.

The system integrates many real-time image processing modules into a complete, compact system. The system incorporates: (1) a custom VLSI low-level image-processing chip set[5][6], (2) a custom VLSI histogram processor and equalization chip set[7][8], (3)

support for an external image processor board, and (4) custom multiplexor ASICs to provide for flexible routing of video signals.

The three image processors can be used, in any order, with commercial A/D-D/A and frame buffer-boards. This system can be reconfigured with the aid of two custom VLSI video crossbar switches, designed and fabricated in a $2\mu m$ SCMOS process. These two application-specific IC designs, fabricated in 132 and 68-pin packages, are fabricated and are fully functional at 10MHz video rates. Four of the 132-pin version and one of the 68-pin version provide 24-bit (color) and 8-bit multiplexing of video busses as required by the chosen image-processing algorithm. Internal pipeline registers ensure 10-MHz throughput, and an external processor can write to the internal configuration registers.

The entire-image processing system has been fabricated on a 9U VME board and is fully functional at video rates. It contains 16 custom ASICs and 2 programmable logic devices (PLDs) for interface logic. A slave VME bus interface provides control of all the image processors and video multiplexer configurations. The board resides in a 21-slot stand-alone card-cage along with a 68020-based CPU card, an Ethernet card, and a custom robot controller card. The board is connected by Ethernet to a Sun workstation and is controlled through a set of X-window front-end tools. Window-based software has been written to reconfigure the processors in any arbitrary fashion.

Chapter 2

System Overview

2.1 Original System

The original image processing system, shown in figure 2.1, contains many sub-modules performing several real-time video tasks while maintaining an overall throughput rate of 10Mhz. The system can be divided into sections according to the task performed: video interface and storage, signal processing, system reconfiguration, software control, and display.

The video interface is performed by a commercial A/D-D/A board, manufactured by Imaging Technologies, Inc. (ITI), that performs the A/D and D/A conversions. The A/D section of the ITI board samples and quantizes the input analog signal (NTSC standard) into its three color (red, green, and blue) components, giving 8 bits of precision per color plane. The D-A section takes 24 bits of video data via three 60-pin connectors and generates the required analog signal to drive the display unit. This ITI board also provides the essential video synchronization signals needed for correct system operation, including vertical blank(vblank), horizontal blank(hblank), and horizontal sync(hsync). The input image signal is acquired through different types of cameras and an analog multiplexor placed before the A/D selects one camera as the input source.

Three commercial frame buffer boards, also designed by ITI, provide storage for one frame of color video data, with one buffer board dedicated to each color plane. In the real-time mode of operation, video data at a maximum rate of 10Mhz can be streamed in from one port while data is streamed out from another port for processing or display(through the D/A). These boards can be used to transfer "frozen" images via the VME bus to the

SUN workstation for documentation or as test images for further non real-time processing. Both the interface board and the frame buffer boards reside in a VME card-cage along with a wire-wrapped video multiplexor board.

The system is reconfigured through the multiplexor board[9] which provides a video interface between an external video processing unit and the boards in the card-cage. The video signal processing for the system is performed by these external units. One of the image processing units[5][6] (a wire wrapped board), designed by Reutz, consists of several custom NMOS chips to perform various image processing tasks including sorting, linear convolution, logical convolution, contour tracing, feature extraction, and limited image recognition. Another board[7][8] (also wire-wrapped) implements histogram and histogram equalization functions. One of the two boards communicates to the multiplexor board[9] through 60-pin video connectors. Due to the limited routing capability of the multiplexor board, only one of image processors can access the boards in the card-cage at any given time.

Three 60-pin connectors are used to communicate between the frame buffers and the multiplexor board. There is no direct connection between the A/D board and the frame buffers. The multiplexor board selects the inputs to the frame buffers from either the attached image processor or the A/D board. Similarly, it also chooses the data going to the D/A from either the processor or the frame buffer. Still frames can be dumped from a SUN workstation through the VME bus onto the frame buffer boards for processing or display. Basically, the attached image processor can be used in any order with commercial A/D-D/A and frame buffer-boards.

In a typical mode of operation, the video data arriving at the multiplexor board is routed to an image processor such as the recognition board or the histogram board. The processed data is then transferred back, via ribbon cables, to the multiplexor board which then routes it to the D/A for display via the frame buffers. The system is reconfigured through the multiplexor board using X-window front-end tools run from a SUN workstation.

2.1.1 Problems With The Original System

The original system includes many boards, and communication is achieved through many video connectors, which are undesirable due to inherent noise problems. To make

things worse, long cables have to be used for video processing units that do not slide into the card-cage and reside far away (such as the image processing/recognition board and the histogram board). More boards and cables also imply more space occupied.

Many of the image processing units, including the multiplexor board, the recognition board, and the histogram board are implemented using wire wrapped technology. These boards are not as physically robust as modern day printed circuit boards and are more prone to noise problems. Minimizing noise is especially critical in this system since the idea of some image processing functions is to reduce noise in images.

Due to the limited processing power of the multiplexor board, only one video processing unit can be used at any given time. With this set-up, a filtered image using the image processing/recognition board cannot be histogrammed in real-time. The processors cannot be easily daisy-chained and connectors have to be often swapped for different image processors to access the interface and memory boards. Also, constant swapping tends to to weaken the boards, which are not very robust to begin with.

A color image can be considered as being represented by three separate monochrome images in the red, green, and blue spaces. The multiplexor board takes 24-bit video data as input and sends 8 bits to the attached image processor for video processing. The multiplexor board limits the image processing capability to just one color plane (hard-wired to be green). Often there is a need to provide more flexibility by supporting processing of different color planes.

2.2 New Integrated System

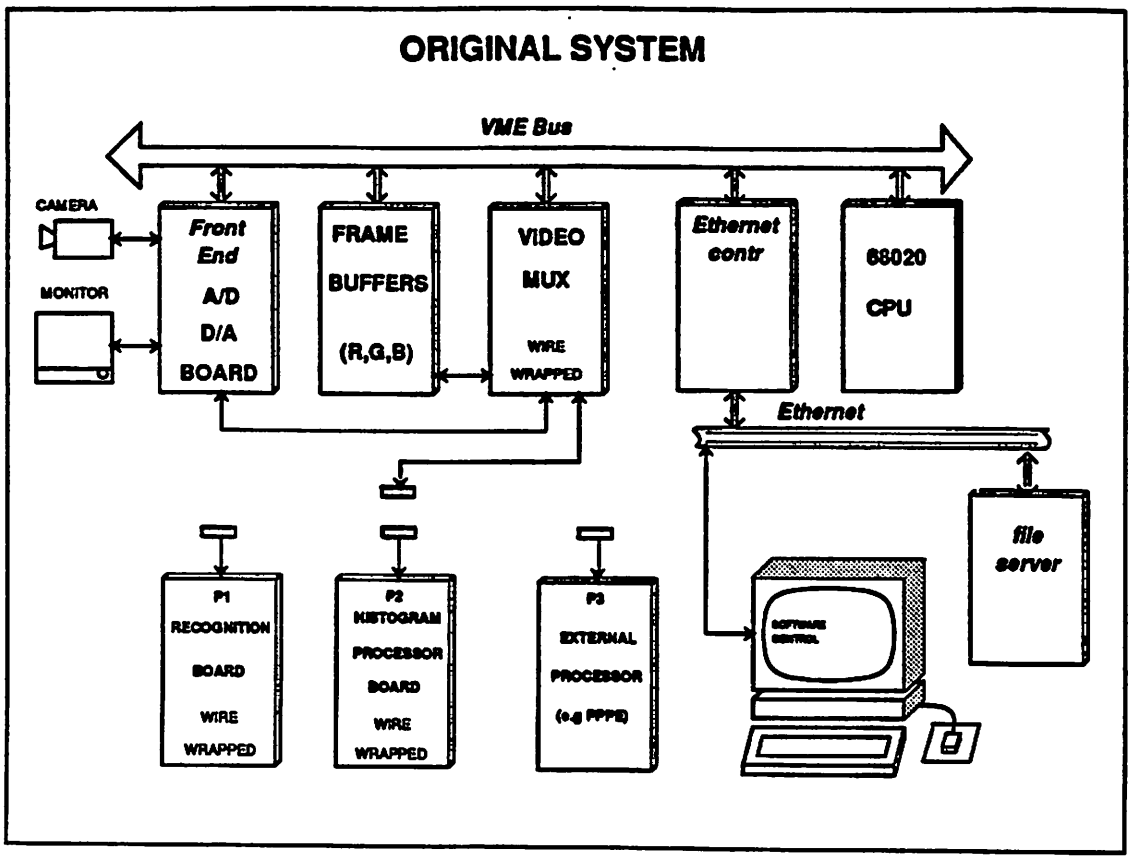
The problems associated with the old image processing system were enough motive to design a more robust and flexible system. The new redesigned system integrates the multiplexor board, the image processing/recognition board, and the histogram processor into one board, and provides support for an external video processor. This single board solution provides a high level of system integration compacting many modules into one slot in the VME card-cage.

Two custom VLSI chips, designed using the LagerIV design environment[10], replaced the existing multiplexor board and provide a lot more flexibility. Any of the three processors can be used in any order with the commercial interface or frame buffer-boards using a set of 5 multiplexor ASICs. Four of these ICs are the 132-pin version PGA's pro-

viding for 24-bit color multiplexing. The fifth ASIC is a 68-pin flat pack providing for 8-bit (one plane) multiplexing. The custom version of the multiplexor is much more compact, enabling it to reside in the new integrated board. This eliminates the connectors between processor and multiplexor board. The three plane multiplexing capability has enabled the processing of any arbitrary plane. These chips are internally pipelined and work at the required 10Mhz rate. An external processor can write to the internal configuration registers in any of these ASICs.

Since two of the image processors are on the board, it has eliminated the need for connectors to communicate video signals between them and the multiplexor. This make the system more robust. The different video processors can be easily cascaded with help of the custom multiplexor chips enabling more functions to be performed on the image data. This added flexibility is very useful and could not have been easily accomplished with the original multiplexor board (only one processor could act on the image data). Extra cables, compatible with the Data cube format, have been added to support an external video processor, such as the Parallel Projection Pipeline Engine (PPPE). This provides full 24-bit color access access to the video setup (interface board, frame buffer boards and the other image processing modules).

Most of the control logic required to operate the various image processing chips were mapped onto PLDs saving significant board area. The first generation video system used a Multi-bus card-cage and the multiplexor board required a Multi-bus to VME adapter. The interface has been redesigned to interface directly to the VME bus, eliminating the need for an adaptor board. Front-end software tools, that are used to reconfigure the system from a SUN workstation, have been modified to reflect the added flexibility of the new system.



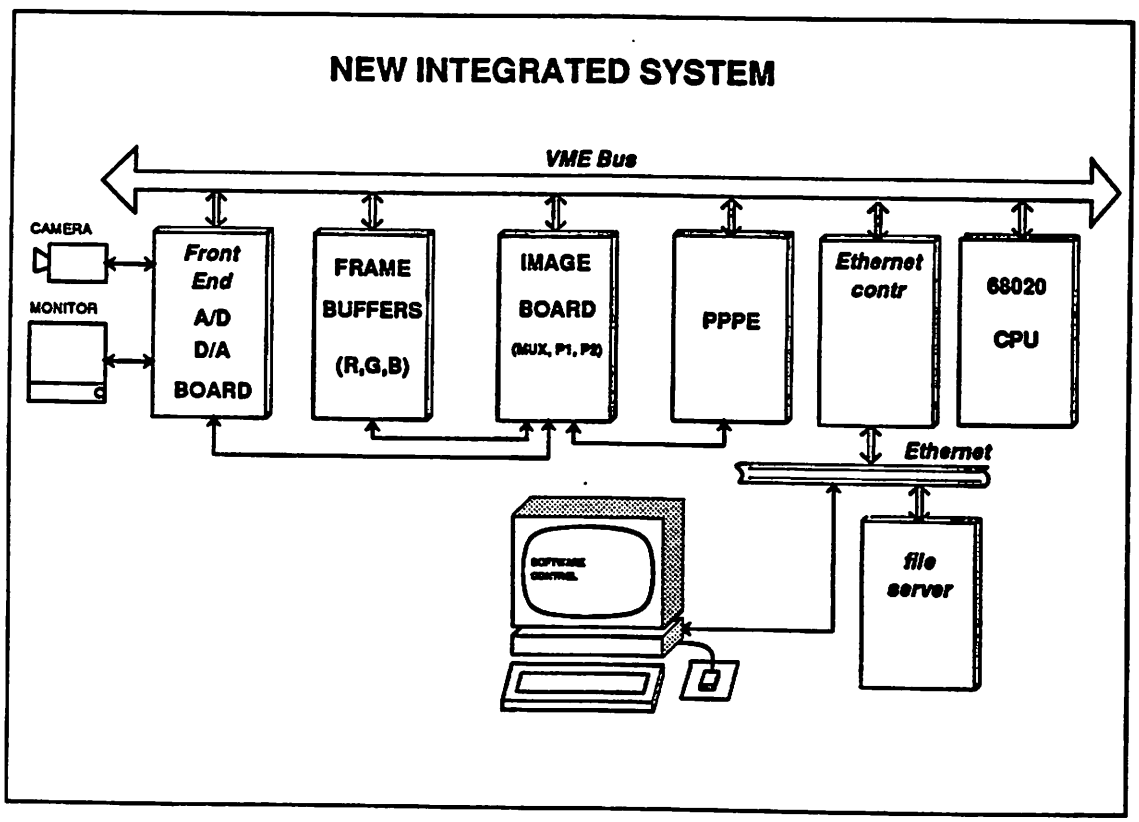


Figure 2.1: Original and Present System

Chapter 3

The Video Multiplexors

This chapter describes the two VLSI video crossbar ASICs that provide flexible routing of image data at video rates(10MHz). The set of two custom video multiplexors were implemented to provide flexible interconnection between the image processors, the interface board and the frame buffer-boards. These chips are critical components for the development of a flexible and compact system that allows interconnection of different modules in an arbitrary fashion.

One of the ASICs provides for three-color channel multiplexing (working on 24 bits of video data) while the other ASIC supports only one color plane (working on 8 bits of video data). One of the three-channel multiplexors replaces the whole first generation multiplexor board while providing additional functionality. The board contains four of the three-channel multiplexor ASICs and one single-channel multiplexor ASIC. Both custom VLSI video crossbar switches were designed using the LagerIV design environment[10] and fabricated using the MOSIS $2\mu m$ scalable CMOS process. The following sections describe the functionality and details of the two multiplexor ASICs.

3.1 Three Channel Multiplexor

The three-channel video cross-bar switch was designed to provide flexible multiplexing for color video data. The video data arriving from the A/D and frame buffers are 24-bits wide (8-bits per color plane). The following subsections describe the functionality and chip details of the three-channel multiplexor.

3.1.1 Functionality

The three-channel video multiplexor (see Figure 3.1 for schematic representation) has two sets of video inputs (48 input pins) and two sets of video outputs (48 output pins). Out of the 48 input pins, 24 pins correspond to input video busses R1SDI, G1SDI, and B1SDI and another 24 correspond to video busses R3SDII, G3SDII, and B3SDII. One set of output busses are labeled R2SDI, G2SDI, and B2SDI (24 bits total) and the other set as R3SDI, G3SDI, and B3SDI (another 24 bits).

The multiplexor is pipelined using a set of level-sensitive registers used in a master-slave configuration. A set of these level-sensitive registers are used at the front end to latch in the video data on the input busses R(G,B)1SDI during the positive phase of the system clock (which is DOTCLOCK_L). Internal buffers are used to provide enough drive to the clock signal. The slave registers are active on the opposite phase of DOTCLOCK_L. The chip requires only one clock input and the other phase is generated by locally inverting the clock. Each datapath register has two clock inputs LOAD and LOADBAR for the true and complement signals. By feeding the LOAD signal of the master register the positive phase of DOTCLOCK_L and the LOAD signal on the slave register the negative phase of DOTCLOCK_L, pipelining is achieved.

The output of the master registers drive multiplexors and output tri-state buffers. The data latched into the pipeline registers are output onto the R(G,B)3SDI busses when the tri-state buffer driving this bus are enabled. The tri-state buffers are enabled or disabled via three control signals (one for each plane) written from the VME interface through a set of standard cell control registers. Latching control data on the chip proved to very area-efficient at the board level. Since the chip contains 12 control bits, every ASIC saves two register chips (such as 74LS74s) at the board level. The tri-state bus permits tying video output busses from different multiplexor chips together. By enabling one of the chips to have its tri-state outputs enabled, a wired-or multiplexor is emulated. There may be some concern with this setup during power-up, before the internal registers are written, when random values appear on the output of these registers.

Any combination of inputs can be routed to any other set of video output busses R(G,B)2SDI. Any one of the six input busses (R(G,B)1SDI , and R(G,B)3SDII) can be output on R2SDI, or G2SDI, or B2SDI. In other words, any output can be derived from any input color, providing for complete mixing of colors. Also, all the output video busses

can derive their outputs from the same color plane. For example, the green input can be routed to all three output busses, R(G,B)2SDI. This feature allows an 8-bit color camera or single plane image processor to drive all three frame buffers. If such data on the frame buffers is used to drive the display through the D/A board, a grey level image will appear.

The multiplexing is achieved through the two-input datapath multiplexor cell. The datapath contains three levels of logic to realize the required flexible multiplexing. The exact configuration of video signals is programmed by control signals written externally via writable control registers. Writing of all control signals takes place during the high-to-low transition of chip select signal (CS_L). Big datapath drivers (using strings of inverters) are used to drive pad drivers. The pad drivers provide the current drive required for highly capacitive output pins and PCB traces. The slave pipeline registers placed before the drivers latch the data onto the output bus during the negative phase of the DOTCLOCK_L.

3.1.2 Chip Details

This ASIC is designed using the LagerIV design environment[10] and fabricated using the MOSIS $2\mu m$ process. The connectivity was specified in a textual format using the structural description language (sdl) and compiled using DMoct. The chip datapath contains multiplexor cells from the dpp library[4] and was compiled using the dpp compiler. Standard cell control logic was implemented to provide external control in configuring the datapath.

The ASIC is packaged in a 132-pin PGA. Out of the 132 pins, it contains 110 signal pins, 4 Vdd and 4 GND pins, and one test pin. The chip is internally pipelined and is functional at the required 10Mhz. The core of the chip is 2789λ * 1777λ. With $\lambda = 1.0\mu m$ for this pwell process, it occupies $4.95mm^2$ of silicon area. The CAD tool "Padroute" was used to route nets from the core to the pads. A photo of the the fabricated chip is shown in Figure 3.2

3.1.3 Chip Simulations And Testing

To test the functionality of the multiplexor, the magic layout was extracted from the pads using DMpost, a post processor, and a simulation file was created. This "sim" file was then used as an input to simulator IRSIM, which uses switch level or linear models to provide approximate timing information. The simulations showed that the chip was fast

THREE COLOR CHANNEL VIDEO MULTIPLEXOR

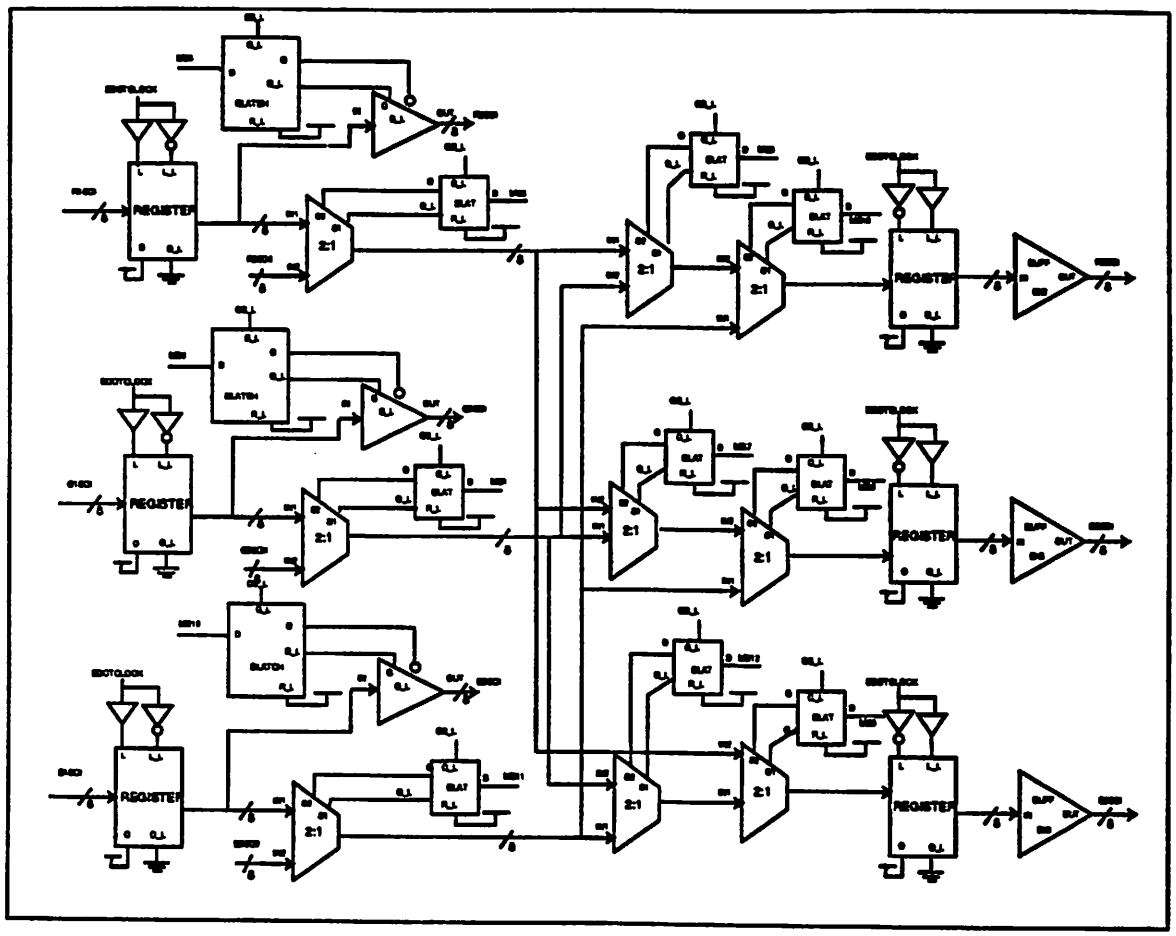


Figure 3.1: Three Channel Cross-bar ASIC Schematic Diagram.

enough at the required 10Mhz.

The chip was tested using the Tektronix DAS 9100 on a generic wire-wrapped test board built to test 132 PGA's. The basic testing strategy was to use the DAS to generate input patterns similar to the ones used in the IRSIM simulations and verify the output against the ones obtained from simulations. The chip was found to be functional at 25Mhz. Better testing equipment is required for testing at higher speeds.

3.2 One Channel Multiplexor

The one-channel multiplexor is highly integrated with the three-channel version. This ASIC supports single-channel (8-bit) multiplexing. The following sections describe the chip in more detail.

3.2.1 Functionality

The one-channel multiplexor (see Figure 3.3 for schematic representation) contains three 8-bit wide input busses(labeled AD_OUT, P_OUT, and FB_OUT) and three 8-bit wide output busses(labeled FB_IN, P_IN, and DA_IN).

Just like the three-channel multiplexor, this ASIC is internally pipelined using datapath registers used in a master-slave configuration. Data on input busses AD_OUT and FB_OUT are latched on the positive phase of the system clock (DOTCLOCK_L). The slave registers are active on the negative phase of DOTCLOCK_L and latch the data onto the output bus when the clock is low. A two-phase clock is emulated internally by swapping the senses on the LOAD signal controlling the pipeline registers.

The chip contains three parallel multiplexing datapaths each working on 2 different sets of inputs. The first path selects between P_OUT and AD_OUT, the second between AD_OUT and FB_OUT, and the third between P_OUT and FB_OUT. In considering the area trade-offs at the board level, it became apparent that implementing two of these multiplexors in one 68-pin package would be optimal. In so doing, some of the internal video busses are shared in common and do not need external I/O pins off-chip.

The datapath is reconfigured through a set of three standard cell registers (edge-triggered). Data is latched in these registers on the high-to-low transition of the chip select signal(CS_L). Big drivers were placed in the datapath to drive the pins and external world.

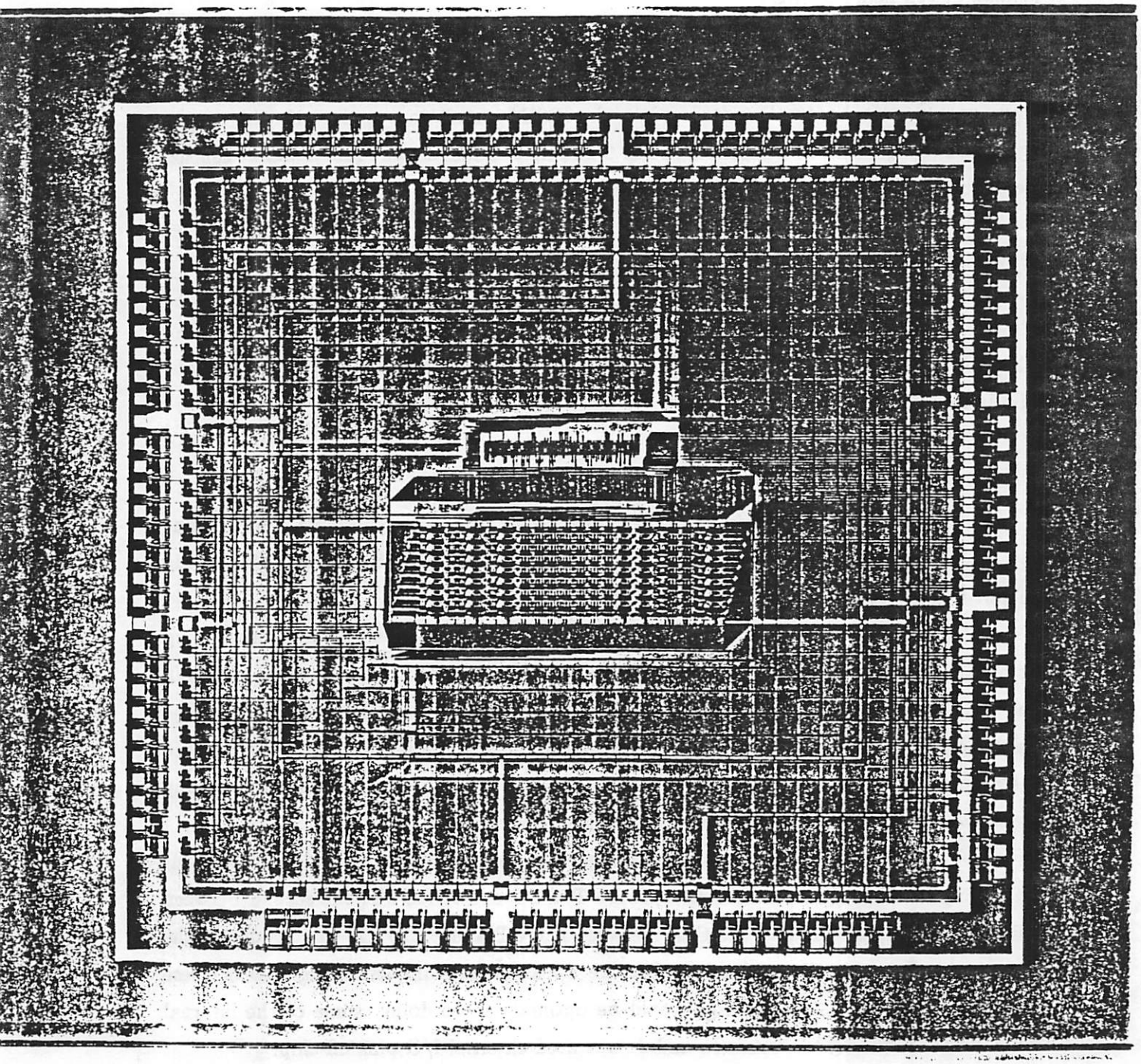


Figure 3.2: Chip Photo of Three-channel Multiplexor.

ONE CHANNEL VIDEO MULTIPLEXOR

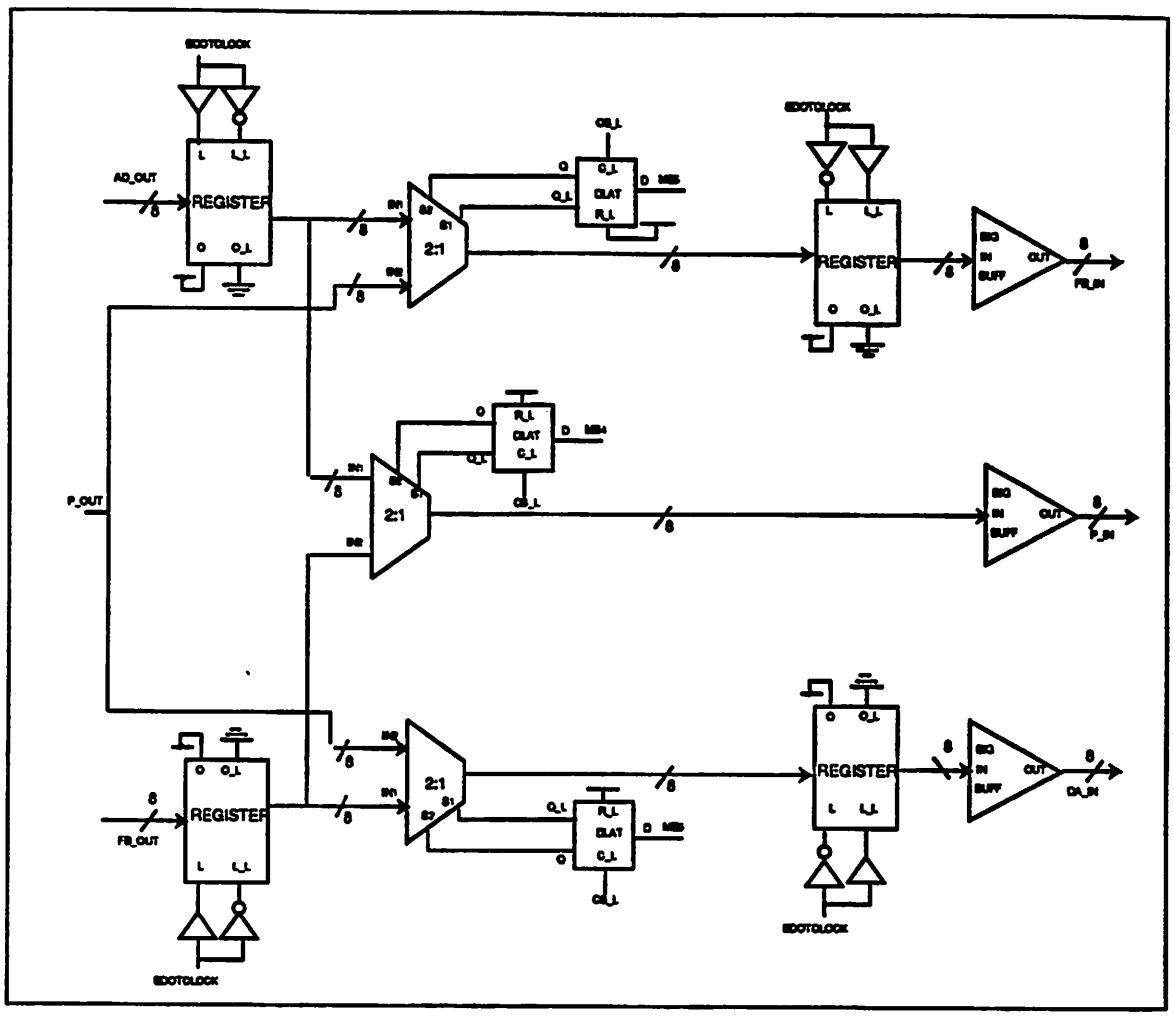


Figure 3.3: One Channel Cross-bar ASIC Schematic Diagram.

3.2.2 Chip Details

This ASIC is also designed using the LagerIV design environment[10] and fabricated using the MOSIS $2\mu m$ scalable CMOS pwell process. The datapath contains multiplexor cells from the dpp library[4] and and uses a standard cell control logic. It is packaged in a 68-pin flat pack. Out of which it contains 55 signal pins, 4 Vdd pins and 3 GND pins and one test pin. Three pins were dedicated for substrate connections.

The chip is internally pipelined and is functional at the required 10Mhz. The core of the chip is 1367λ * 1033λ and hence occupies $1.4mm^2$ of silicon area. A photo of the fabricated chip is shown in Figure 3.4.

3.2.3 Chip Simulations And Testing

The layout of the one-channel multiplexor was also extracted from the pads using DMpost and simulated and verified using IRSIM. Various input patterns were used to test functionality and speed. The simulations showed that the chip was logically functional and fast enough at the required 10Mhz.

The testing of this chip was done on the new image processing board. This proved to be easier than building a separate test board. Test images generated off-line were dumped into the framebuffers and routed via the one-channel multiplexor. By reconfiguring the chip through the VME bus, while observing and comparing the output on the monitor, the functionality of the chip was verified. The speed was of lesser concern since it has fewer levels of logic in the datapath than the three-channel multiplexor and it was clear that is was fast enough for the application.

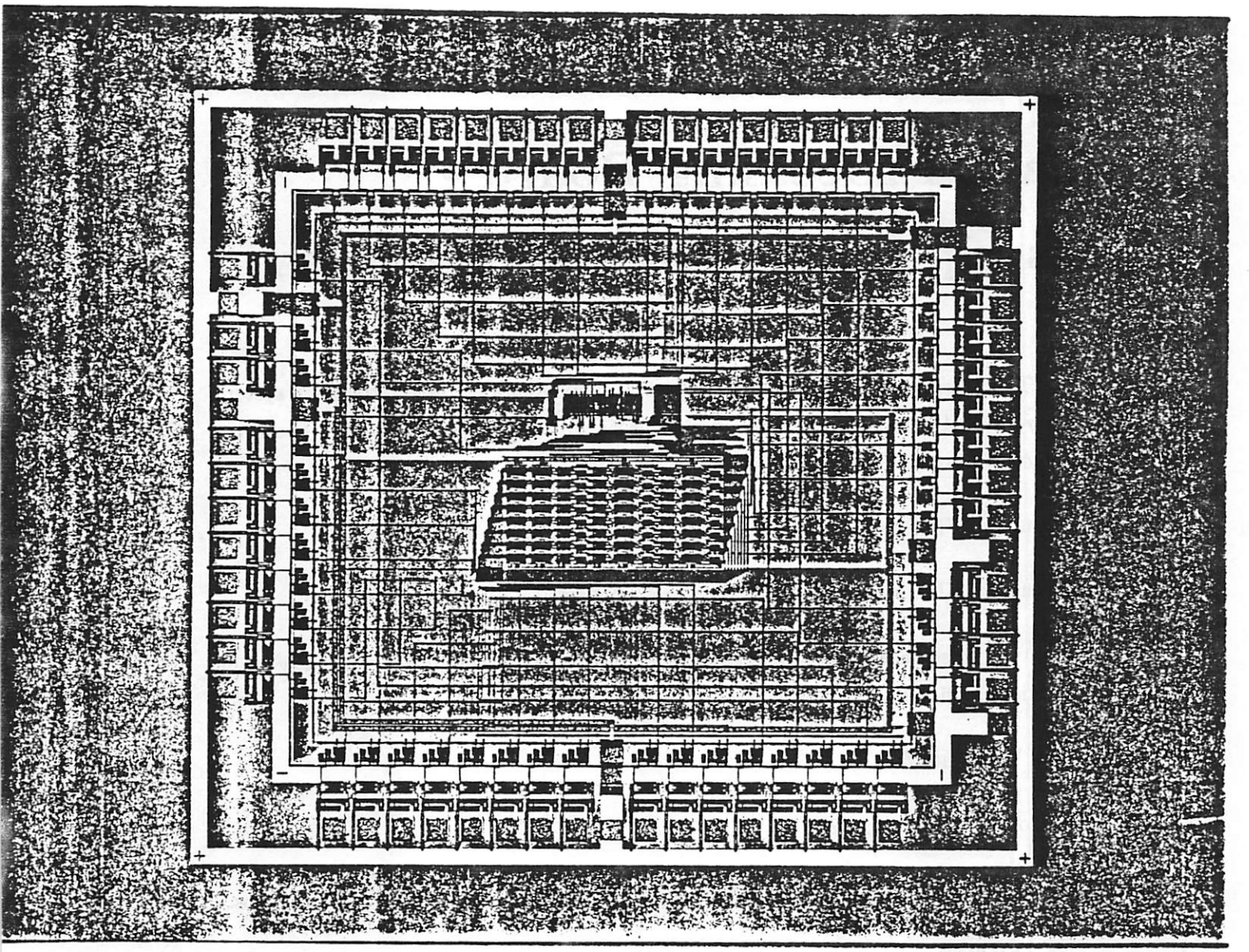


Figure 3.4: Chip Photo of One-channel Multiplexor.

Chapter 4

Image Processing Board - Description Of Various Modules

The image processing board integrates many modules into a compact system. This board is VME based and resides in a triple height 21-slot card-cage. The board can be divided into functional modules including:

- A custom VLSI image processing and recognition chip set module.
- A custom VLSI histogram processing and equalization chip set module.
- Custom video multiplexor (color and greyscale) module, supporting two-way 24-bit communication to an external processor.
- The VME interface.
- A clock generator module.
- A control module generating required control signals to the image processing chips.

The flexible image processing board (See Figure 4.3 For a Block Diagram) sits on the VME card-cage along with other commercial interface (A/D and D/A boards) and frame buffer boards. The card-cage is a stand-alone system with a single-board computer based on Motorola's 68020 CPU chip and controls all operations over the VME bus. It runs "VxWorks", which is a real-time unix-like operating system. Software written on the SUNs

can be compiled and down-loaded onto this system. The card-cage also has a commercial ethernet controller card that acts as a slave processor on the VME bus. Image files can be dumped to and from the SUN workstation through the ethernet via the network file server. One important application is that test images can be used for debugging the hardware.

4.1 Description Of Modules

This section will overview the various modules on the board and describe the function they play in the overall system.

4.1.1 Video Interface To The Board

The board interfaces to the A/D-D/A board, and the three frame buffers through video connectors. Three 60-pin ITI connectors, one for each color plane, are used to pass signals to and from the A/D board, with each cable carrying both an input and an output eight-bit video bus. These connecters also pass synchronization signals, such as DOT-CLOCK_L, VBLANK_L, and HBLANK_L, required for system operation. The complete set of signals with pin mappings is compiled in Appendix C. Each frame buffer is also connected to the image processing board via 60-pin connectors. The board also provides external port access to an image processor. This is done via six 26-pin digital video connectors using the "Data cube" "Maxbus" video format. Three of these provide red, green, and blue input channels, and the other three serve as output channels to other image processor boards in the card-cage.

Video drivers, mostly 74AS244, were used for signals that are driving highly capacitive video bus wires. In one case, the AS family of drivers resulted in a high dV/dt and hence high mutual inductance, causing erroneous data on the other side of the connector. This cross talk between adjacent bits was reduced by replacing the 74AS244 chip with a 74LS244.

The video busses communicate to the video processors through the multiplexor chips. Since the incoming signals drive only one multiplexor, buffers were not inserted, saving area.

4.1.2 Clock Generator Module

Most of the image processing chips required a two-phase non-overlapping clock signals. The original system achieved this by using a pair of cross coupled NOR gates (since NOR gate ensure non-overlapping signals) with one input being a buffered version of DOTCLOCK_L and the other is an inverted version of DOTCLOCK_L. These chips ran off 7V power supplies to provide fast rise time. With TTL NOR gates this translated to around 5V swing on the clock lines. The present board eliminates the need for an extra power supply by replacing the slow TTL NOR gates with fast CMOS NOR gates that provide rail to rail swing (74HC02's). Since the HC family requires CMOS input levels, the buffers and inverters driving the HC chips are from the ACT family that convert TTL input levels(from DOTCLOCK_L) to CMOS levels. The clocks have a frequency of 10Mhz and a duty cycle of 55-45%.

4.1.3 Image Processing/Recognition Module

The board has two image processing modules. The first of these modules contains a set of 8 custom NMOS chips[5][6] to perform real-time image processing tasks (as shown in Figure 4.2). The chips include a 3x3 linear convolver, a 3x3 sorting filter, a 7x7 logical convolver, a contour tracer, a feature extractor, a look-up table ROM, and 2 post processors for the linear convolver. Most of the chips are cascaded (see Fig 4.2) and clocked with PHI1 and PHI2. These are non-overlapping clocks as described in the clock generation module. The feature extractor and contour tracer have a more complicated interconnection. The input video signals to this module are latched in through an 8-bit wide register clocked by PHI1 and the output video signals are latched out through another 8-bit wide register clocked on the opposite phase.

Low rate control signals from the host are latched by a bank of seven 74LS374 registers. These signals configure the chips internally for different modes of operation. For example, the convolver chip has many convolution kernels stored internally and one of them can be chosen at a time. Similarly, low rate data is read out to the VME bus through a bank of four 74LS244 drivers. The VME interface generates control signals to select these latches and drivers.

The other control signals required by the chips are generated on board and described in the control module section.

CLOCK GENERATION CIRCUITRY - OLD AND NEW

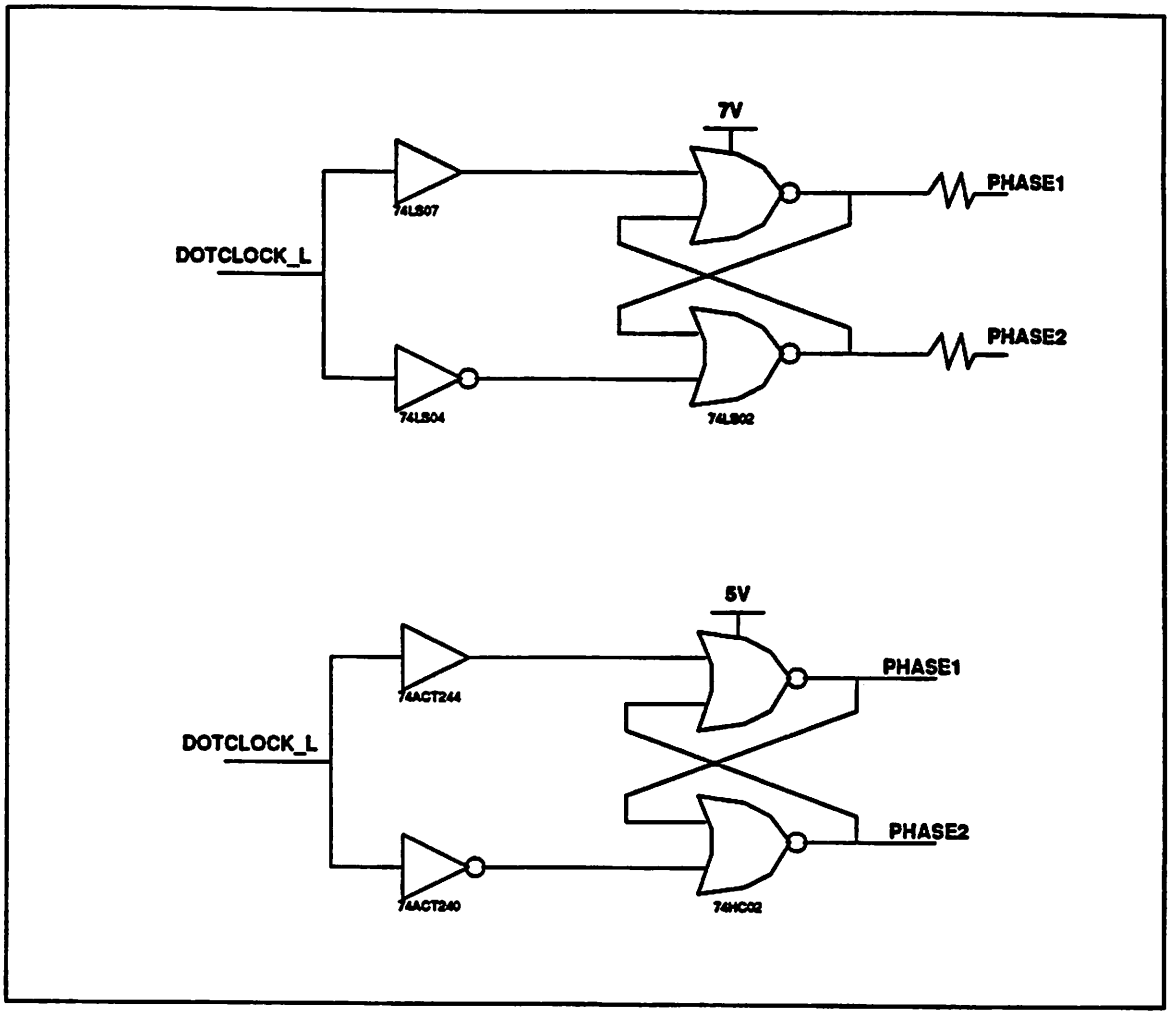


Figure 4.1: Clock Generation Circuitry - Old and New.

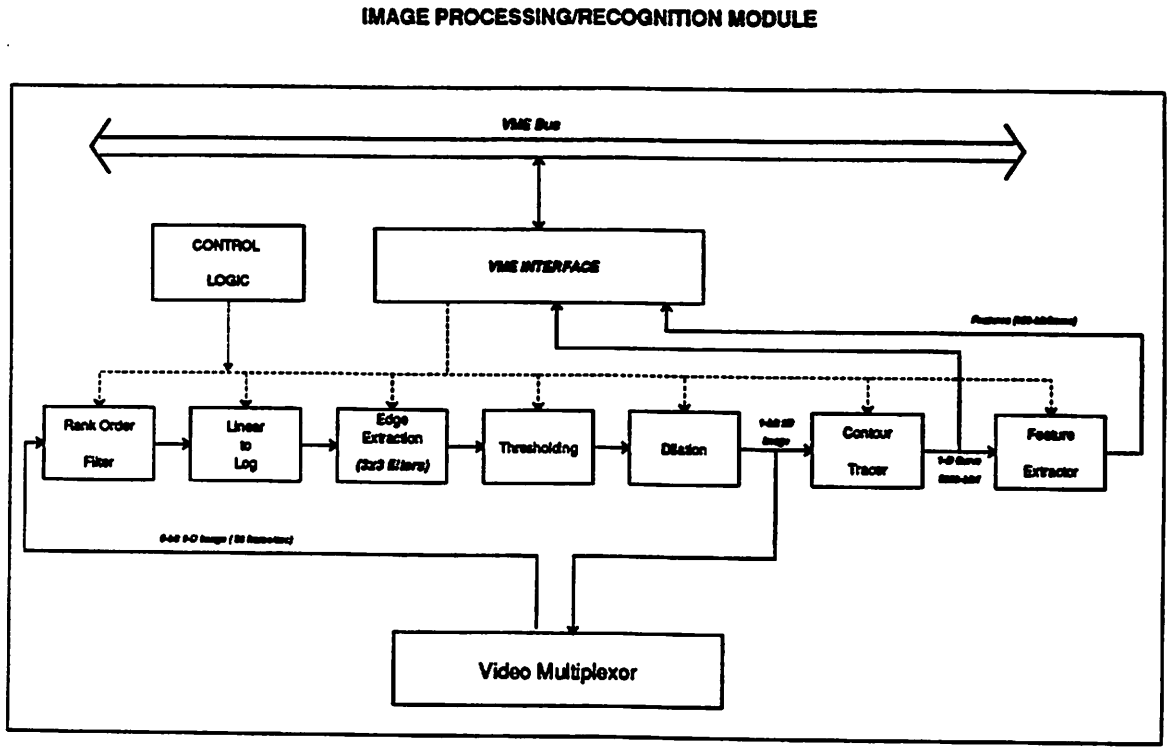


Figure 4.2: Image Processing/Recognition Module.

4.1.4 Histogram Processor and Equalization module

This module contains custom chips to perform real-time histogram and histogram equalization[7] . These chips are designed using a MOSIS NMOS technology. Histogram estimates the pixel intensity distribution over a selected portion of an image. The equalization chip modifies the contrast of an image by changing the pixel intensities of an image to result in an equal distribution of grey levels in the image. The video input to this module is latched in through an 8-bit register and the video out is latched out through another register. The control signals to the chips are provided by dip switches that set the values to zero or one depending on whether they are closed or open. The chips are clocked using the non-overlapping two-phase clocks PHI1 and PHI2 generated by the clock generation circuitry.

4.1.5 VME Interface Module

The VME interface (shown in Figure 4.3) provides the communication between the image processors and the VME bus host. The VME based card-cage has a 68020-based single-board computer running Vxworks, a real-time Unix-like operating system. This acts as a system controller and arbitrates all transactions on the VME bus. The VME interface performs the following important functions:

- Decodes sixteen address bits (A[9-25], AM[0-5], A[6-8], and LWORD_L) and determines which processor the VME master is communicating with.
- Allows data from the VME to be written to all the multiplexors and image processing modules over 12-bits on the DATA bus D[0-11].
- Allows low-rate data from the image processors to be read to the VME from the processors.

The VME interface provides communication between three banks on the board and the VME bus. Out of these, two banks are writable (five multiplexor chips and four 74LS374 latches) and the other is readable (seven 74LS244 drivers). The interface not only selects the particular bank, but it also enables one particular chip in the bank.

A set of drivers (74LS244 and 74LS645) are used to buffer address lines from the VME bus (in accordance to the VME electrical specifications). The VME interface uses

a two-level decoding strategy. The first level performs the decoding of A[9-15],AM[0-5] and LWORD_L and is implemented using an Altera EP610 PLD. This basically selects the particular bank (one among three). The second level of decoding (A[6-8]) strobes one particular chip in the bank. This stage is implemented using 74LS138 decoder chips.

Another EP610 PLD handles the synchronization of data transfer. It monitors the VME data strobe (DS_L) and write control signals from the VME bus and generates the acknowledge signal (DTACK_L). It also controls the strobe lines on the vector decoder chips (the second stage of decoding).

Table 4.1 lists the exact address space of the different processors that communicate with the VME interface.

Instance Name	Address	Reference Designator	Type
pip_xx374_2	0xfffc00	U50	Writable
pip_xx374_3	0xfffc40	U51	Writable
pip_xx374_4	0xfffc80	U52	Writable
pip_xx374_5	0xfffcc0	U53	Writable
pip_xx374_6	0xfffd00	U54	Writable
pip_xx374_7	0xfffd40	U55	Writable
pip_xx374_8	0xfffd80	U56	Writable
board_MUXFB1	0xfffe00	U1	Writable
board_MUXFB2	0xfffe40	U2	Writable
board_MUXVP1	0xfffe80	U3	Writable
board_MUXVP2	0xfffec0	U4	Writable
board_MUXPI	0xffff00	U5	Writable
pip_xx244_1	0xfffa00	U43	Readable
pip_xx244_2	0xfffa40	U44	Readable
pip_xx244_3	0xfffa80	U45	Readable
pip_xx244_4	0xfffac0	U46	Readable

Table 4.1: Address Space for Various Processors and Multiplexors

4.1.6 Multiplexor Module

The board contains four color video cross-bar switches and one single-plane switch. These chips provide flexible routing of video data between processors and interface boards. Using these chips we can reconfigure the system to utilize the processors in any order with the interface and storage boards. It also permits the cascading of different processors. The complete functionality of the chips were discussed in Chapter 3.

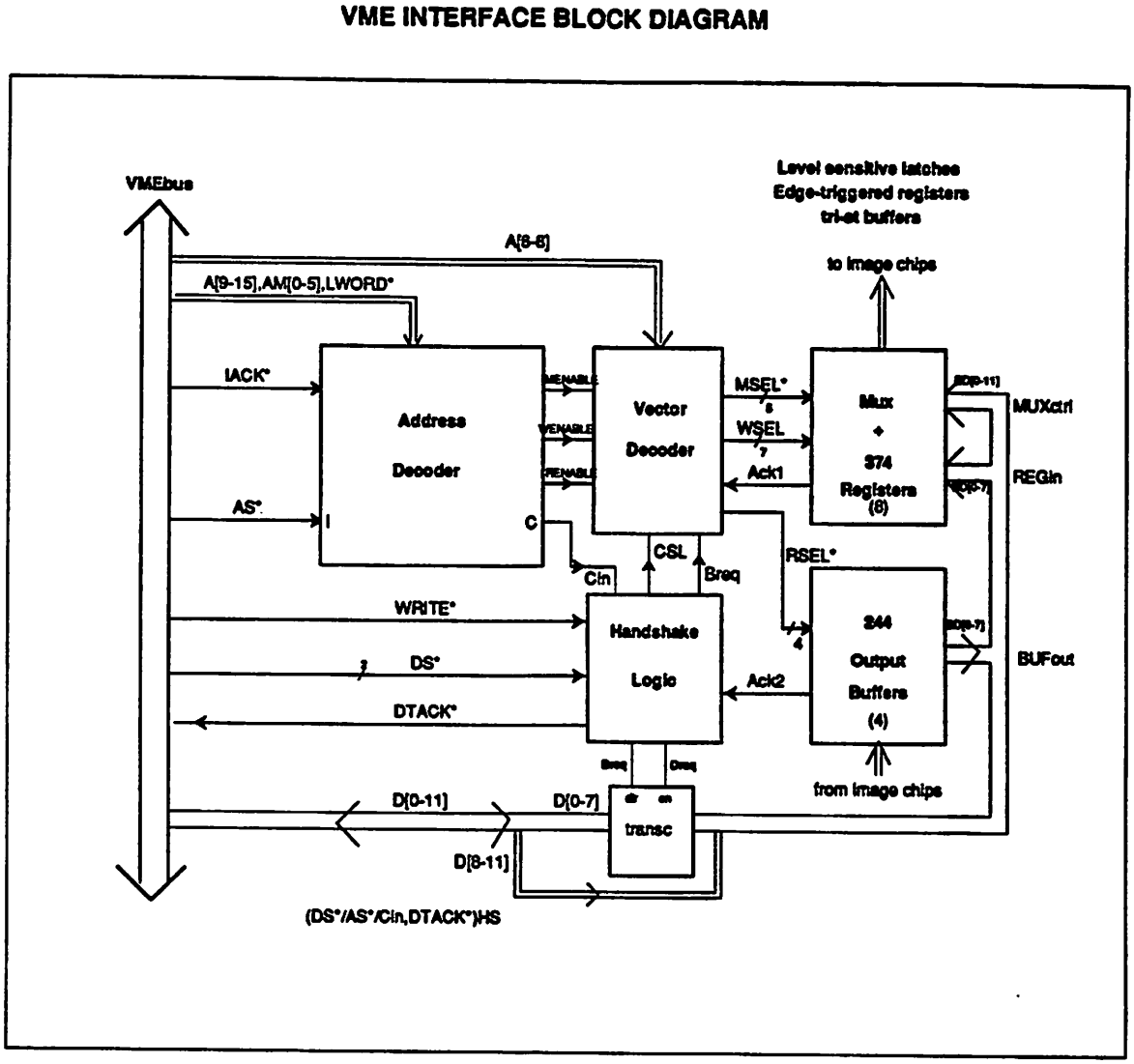


Figure 4.3: VME Interface Block Diagram.

4.1.7 Controller Module

The control signals required by many of the image processing chips were produced using many TTL chips in the original system. In the present system, all the glue logic for the control signals are programmed and densely packed into a 132-pin Altera EP1800 PLD. The logic was simulated using the THOR simulator and approximate timing analysis was used to make sure that the logic meets the speed requirements.

4.2 Interconnection of Modules

The interconnection of all the modules on a single PCB resulted in a very flexible image processing system. A block diagram of the board is shown in Figure 4.4.

The video connectors interface to the outside world (the interface boards and the storage boards). Each of the image processing modules (including the external processor) are abstracted as blocks in the block diagram with video-in and video-out busses and control signals connected to each block. A set of five multiplexor chips are used to provide flexible interconnection between the modules. The input video signals (24-bit color) from the A/D board goes into a three channel multiplexor. The source of the input to the frame buffers can either be from the A/D or an image processor depending on how this multiplexor is configured. The outputs from the frame buffer boards come in to another three-channel multiplexor on the board. The tri-state output busses of these two multiplexors are tied together and act as the input to another three-channel multiplexor controlling the data to the external video processor. The output from the off-board expansion processor is routed through another three-channel multiplexor chip.

Four of the multiplexors are thus used in controlling the I/O of the A/D, frame buffers, and off board expansion processor. The single-channel multiplexor handles the routing for the two single-plane image processors. The two blocks labeled 1CHMUX are physically only one chip. With this setup, the video data can be very flexibly routed through different processors and hence allows processors to be easily daisy-chained.

The VME interface module interfaces to the VME bus. It produces control signals that select particular processors and multiplexor chips and writes control words that configure them in the appropriate modes of operation. Other control signals are generated by the EP1800 PLD and are distributed to the image processors.

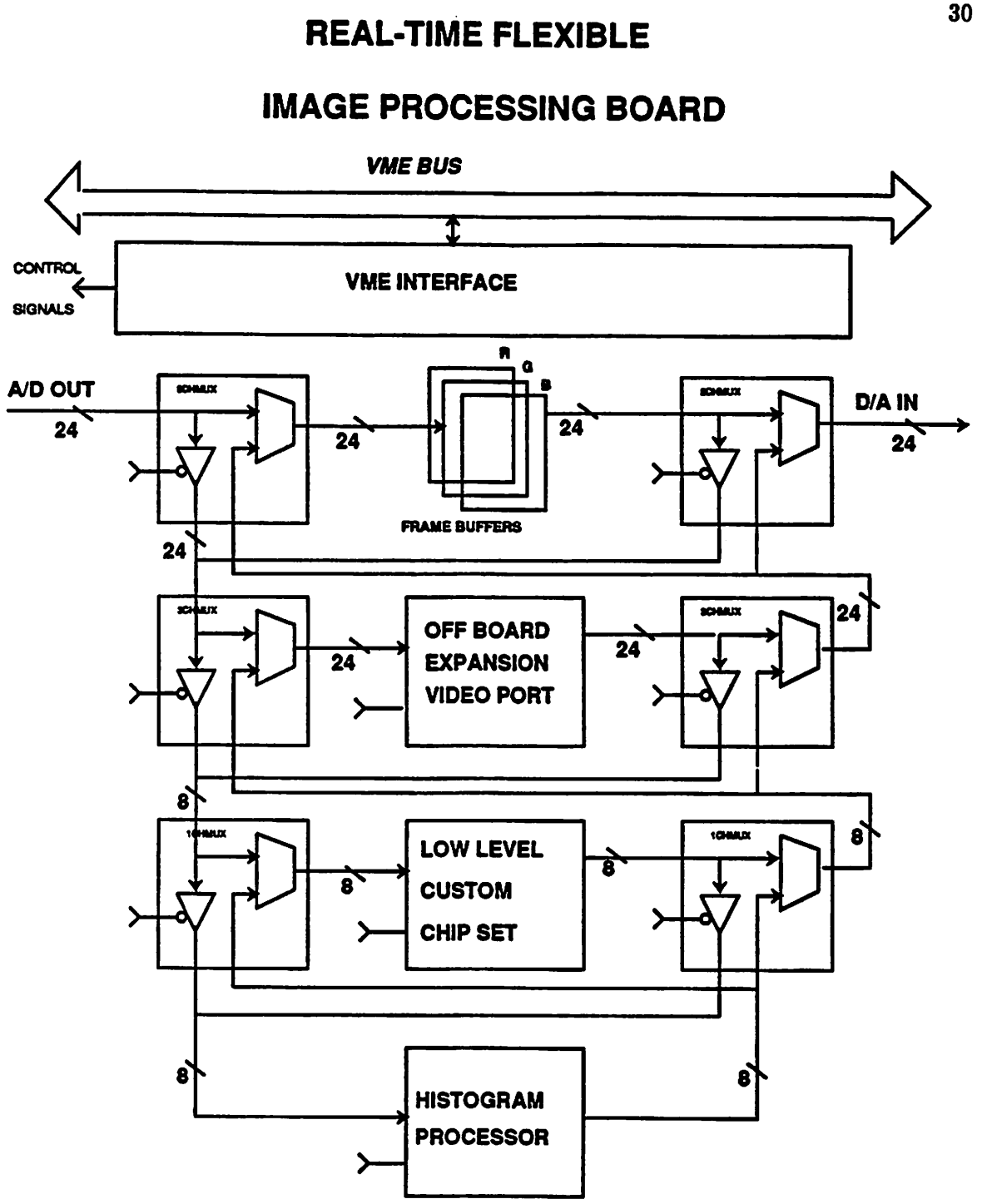


Figure 4.4: Block Diagram of The Image Processing Board.

The image lab has a graphical user interface. When the "image-lab" program is invoked from a SUN workstation, an X window pops up with boxes representing different processors (as shown in Figure 4.5). Selecting a box by clicking on it, opens more windows in some cases. For example the filter chips have many convolution kernels and a particular one can be chosen by clicking on a box. The graphical interface also allows the reconfiguration of the data flow through the multiplexors.

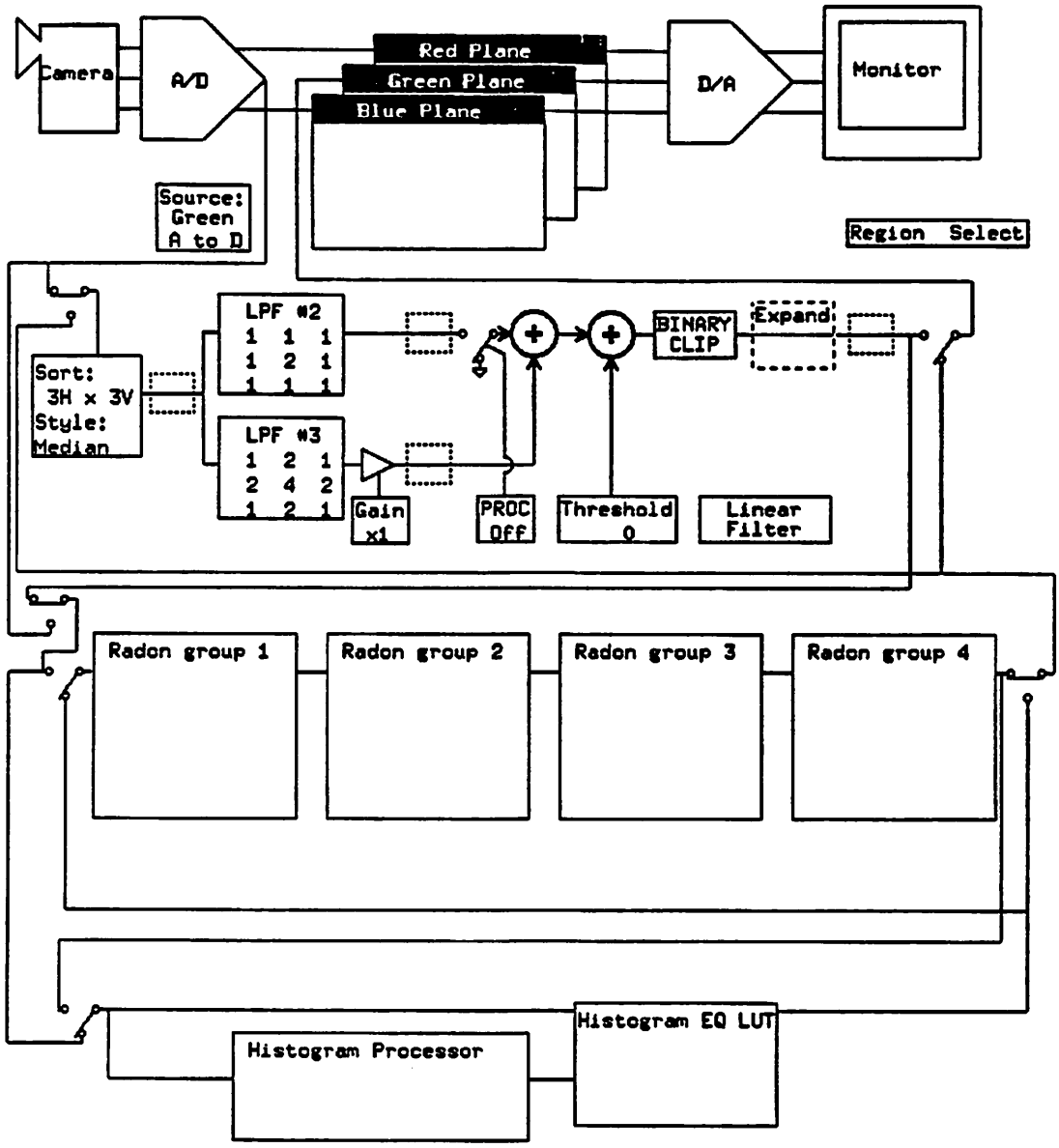


Figure 4.5: Software Front-end Tools.

Chapter 5

Design Cycle And Board Specifics

This chapter outlines the design cycle involved in the design of the 8-layer 9U VME application specific board. This includes net-list entry, simulations, and placement and routing. This chapter also highlights the board specifics and the strategy used for testing.

5.1 Design Process

The board was mainly designed using the LagerIV design environment. A commercial package designed by Racal-Redac was also used in the final phases of the design. The following sections describe the design process and CAD support in more detail.

5.1.1 Design Entry And Simulation

The connectivity information was entered an textual format using the structural description language (sdl). This is preferred over schematic entry. Simulation of the design can be achieved using the THOR functional simulator. Once the design was entered, DMoct was used to generate its representation in the "OCT" database ("SMVs" and "SIVs"). Sdl format is also very useful in visualization of the design in a hierarchical fashion. For example, if three color-plane processing capability were to be incorporated into the image processing module as future enhancement, it can be easily done by instantiating the image processing module (written as one sdl file) three times.

THOR models for the various leaf-cells were written and MakeThorSim was used to generate input files required by the THOR simulator. The design was simulated mainly

for checking the connectivity information.

5.1.2 Place And Route

A commercial package was used for placement and routing. This software requires the net-list to be in a special format called "rinf". The CAD tool "Oct2rinf" extracts net-list information from the "SIV" representation and generates a rinf file, called "design.frs", which is the input to the Visula PCB tool. Oct2rinf also generates an ASCII report file, called "design.rep", which has part/net and chip count information of the board.

The rinf file was loaded into the PCB tool and each part was manually placed. Oct2rinf provides the feature of being able to include the placement information in the sdl file. This way the placement has to be done only once manually and the placement information can be back-annotated into the sdl file. The exact placement information is obtained by dumping a "design.frp" file that reflects design changes. This feature is especially useful when there are minor design changes and and a complete manual placement is undesirable. Hence, very little time is spent in the layout. A trick employed to make the initial placement faster was to create a dummy set of sdl files that have no nets but the same parts as the real design. This way the parts could be moved around faster in the PCB tool since it does not have any nets.

Once the design was simulated and placed, the Racal-Redac router was used to route this 8-layer board (6 signal-layers). After 100% routing, post-processing operations such as mitering and fattening were performed. At each stage of routing, error-checking was performed to fix any design rule violations. Gerber format files were generated and films were produced using these files. After visually checking the film for obvious shorts and other errors, the film was shipped off to the PCB fabrication house.

5.2 Board Specifics

The board is triple-height 9U and resides in a 21-slot stand-alone card-cage. The board contains a total of 16 custom ASICs and three programmable logic devices for interface and control logic. The logic for the PLDs were entered schematically and the PLDs were programmed using the Altera software on a DOS PC. The board contains a total of 12 connectors to communicate video signals and a total of 70 ICs. The ICs sockets were inserted and then the board was wave soldered.

Out of the 8-layers, 6 layers are signal layers and one layer is dedicated to each power and ground. This is to reduce noise in the system. Bypass capacitors were also inserted between power and ground for each IC in order to reduce noise. Two separate proto-type playgrounds (holes without any connections) were placed for testing purposes. Also a few unconnected 20-pin DIP's were placed in unused areas to provide spare parts in case of design errors. A photo of the board is shown in Figure 5.1.

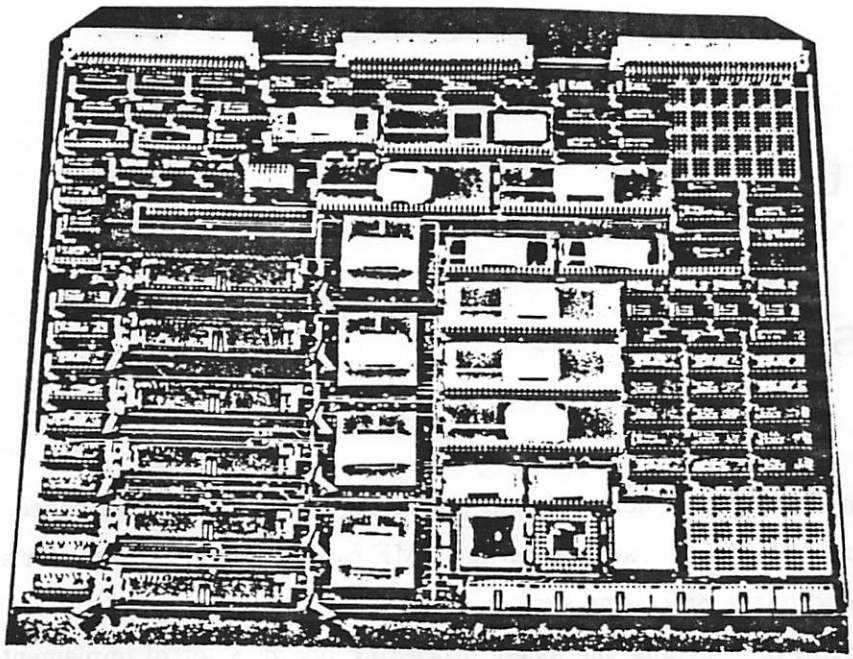
5.3 Testing

The board was tested module by module. The method of using software for debugging hardware was repeatedly used.

First, the VME interface chips were inserted and tested. Different test vectors inputs were written into the writable registers on the board and read back through readable registers. Then various registers were addressed and appropriate strobe signals were probed to check the functionality of the interface.

Next the multiplexors were tested. Test pattern raster files generated using "C" code were dumped onto the frame buffers and routed through the multiplexor ASICs by writing appropriate control words into them. All the clock generators were probed to make sure they produce non-overlapping two-phase signals. Once the functionality of the multiplexors were established, the image processing chips were inserted and video data was routed through them. At this point the software, originally written for the original system was rewritten to reflect new additions and flexibility.

Most of the critical signals were easily accessible because of the headers attached to them for testing purposes.



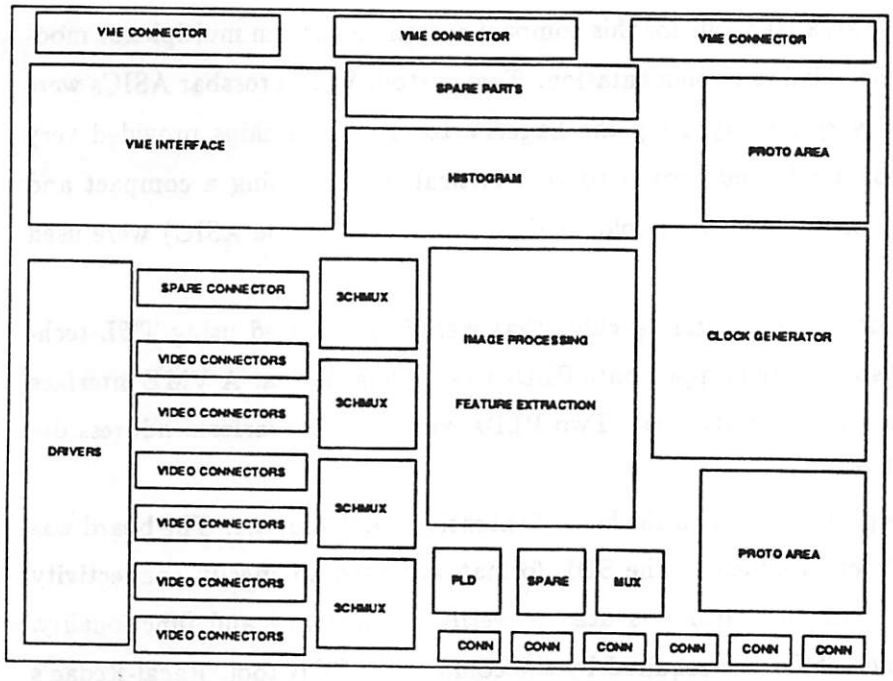


Figure 5.1: Top View of Image Processing Board.

Chapter 6

Conclusions

The design of a real-time flexible image processing board has been described. This functional 8-layer printed circuit board sits on the VME card-cage and provides for a very flexible and integrated image processing system.

In the process of designing the image processing board, a lot of implementation issues were dealt with. A top-down design approach was practiced. First, crucial high level decisions were made and a block level interconnection diagram produced. Given the block level description, each block was then designed.

Considering the area tradeoffs for this compact system, a custom multiplexor module was preferred over the existing implementation. Two custom VLSI crossbar ASICs were designed and fabricated very quickly using the LagerIV tools. These chips provided very flexible routing of video signals and proved to very critical in developing a compact and flexible system. A total of five ASICs (4 color ASICs and one greyscale ASIC) were used on the board.

Control logic for image processing chips that were implemented using TTL technology in the original system were mapped onto PLDs to save board area. A VME interface was designed to interface to the VME bus. Two PLDs were used to perform address decoding and handshake logic.

The board was prototyped with the help of extensive CAD support. The board was designed in the LagerIV environment. The SDL format was used to specify connectivity information, and the THOR simulator was used to verify connectivity and functionality. Oct2rinf produced the netlist format required by the commercial PCB tool. Racal-Redac's PCB software package was used for place and route. Altera's APLUS package was used for

schematic capture and design of the PLDs. The CAD tools proved to be invaluable in the effort to prototype the system very quickly.

This system is capable of real-time image processing and image recognition. This feature can be used in tracking applications and hence can be used to integrate vision and robotics.

A future generation board can be built with the capability of processing real-time color images. This would be very simple in terms of design effort due to the modular design approach used in the design of this system. The image processor module would have to be duplicated three times (for three color planes) by instantiating it three times in a top level sdl file.

Bibliography

[1] Anil K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, NJ, 1989.

[2] J.S. Lim, "Two-Dimensional Signal and Image Processing", Prentice-Hall Inc., 1990.

[3] R.C. Gonzalez, and P. Wintz, *Digital Image Processing*, Addison-Wesley, MA, 1987.

[4] R.W. Brodersen et. al., "LagerIV Cell Library Documentation", Electronics Research Laboratory, University of California, Berkeley, June 23, 1988.

[5] P.A. Ruetz, R.W. Brodersen, "Architectures and Design Techniques for Real-Time Image-Processing IC's", *IEEE Journal of Solid-State Circuits*, Vol. SC-22, pp. 233-250, April, 1987, pp. 233-250.

[6] P.A. Ruetz, "Architectures and Design Techniques for Real-Time Image-Processing ICs", Memorandum No. UCB/ERL M86/37.

[7] B.C. Richards, A. Sherstinsky, R.W. Brodersen, "A Parameterized VLSI Video-Rate Histogram Processor", *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, Dallas, April 1987.

[8] B.C. Richards, "Design of a Video Histogrammer Using Automated Layout Tools", Memorandum No. UCB/ERL M86/38.

[9] Julian, "Video Signal Multiplexing Board", UCB/ERL report.

[10] C.B.Shung, R.Jain, K. Rimey, R.W.Brodersen, E.Wang, M.B.Srivas tava, B.Richards, E. Lettang, S.K.Azim, P.N.Hilfinger, J.Rabaey, "An Integrated CAD System for Algorithmic-Specific IC Design", Publication Pending.

Appendix A

Chip Parameter Files and Bonding Diagrams

THREE-CHANNEL_MUX.par

```
(# 8)
(wpads
 (("in2_2_Ou" "R1SDI[0]" "" "" "" "")
  ("in2_2_Ou" "R1SDI[1]" "" "" "" "")
  ("in2_2_Ou" "R1SDI[2]" "" "" "" "")
  ("in2_2_Ou" "R1SDI[3]" "" "" "" "")
  ("in2_2_Ou" "R1SDI[4]" "" "" "" "")
  ("in2_2_Ou" "R1SDI[5]" "" "" "" "")
  ("in2_2_Ou" "R1SDI[6]" "" "" "" "")
  ("in2_2_Ou" "R1SDI[7]" "" "" "" "")
  ("vdd_2_Ou" "Vdd" "" "" "" "")
  ("in2_2_Ou" "G1SDI[0]" "" "" "" "")
  ("in2_2_Ou" "G1SDI[1]" "" "" "" "")
  ("in2_2_Ou" "G1SDI[2]" "" "" "" "")
  ("in2_2_Ou" "G1SDI[3]" "" "" "" "")
  ("in2_2_Ou" "G1SDI[4]" "" "" "" "")
  ("in2_2_Ou" "G1SDI[5]" "" "" "" "")
  ("in2_2_Ou" "G1SDI[6]" "" "" "" "")
  ("in2_2_Ou" "G1SDI[7]" "" "" "" "")
  ("gnd_2_Ou" "GND" "" "" "" "")
  ("in2_2_Ou" "B1SDI[0]" "" "" "" "")
  ("in2_2_Ou" "B1SDI[1]" "" "" "" "")
  ("in2_2_Ou" "B1SDI[2]" "" "" "" "")
  ("in2_2_Ou" "B1SDI[3]" "" "" "" "")
  ("in2_2_Ou" "B1SDI[4]" "" "" "" "")
  ("in2_2_Ou" "B1SDI[5]" "" "" "" "")
  ("in2_2_Ou" "B1SDI[6]" "" "" "" "")
  ("in2_2_Ou" "B1SDI[7]" "" "" "" "")
  ("in2_2_Ou" "MB4" "" "" "" "")
  ("in2_2_Ou" "MBO" "" "" "" "")
  ("in2_2_Ou" "MB10" "" "" "" "")
  ("in2_2_Ou" "CS_L" "" "" "" "")
  ("in2_2_Ou" "BDOTCLOCK" "" "" "" "")))
(npads
 (("in2_2_Ou" "MB13" "" "" "" "")
  ("io_2_Ou"  "R3SDI[0]" "" "MB4Q_L" "" "")
  ("io_2_Ou" "R3SDI[1]" "" "MB4Q_L" "" "")
  ("io_2_Ou" "R3SDI[2]" "" "MB4Q_L" "" "")
  ("io_2_Ou"  "R3SDI[3]" "" "MB4Q_L" "" "")
  ("io_2_Ou"  "R3SDI[4]" "" "MB4Q_L" "" "")
  ("io_2_Ou"  "R3SDI[5]" "" "MB4Q_L" "" "")
  ("io_2_Ou"  "R3SDI[6]" "" "MB4Q_L" "" "")
  ("io_2_Ou"  "R3SDI[7]" "" "MB4Q_L" "" "")
  ("vdd_2_Ou" "Vdd" "" "" "" "")
  ("io_2_Ou"  "G3SDI[0]" "" "MBOQ_L" "" "")
  ("io_2_Ou"  "G3SDI[1]" "" "MBOQ_L" "" "")
  ("io_2_Ou"  "G3SDI[2]" "" "MBOQ_L" "" "")
  ("io_2_Ou"  "G3SDI[3]" "" "MBOQ_L" "" "")
  ("io_2_Ou"  "G3SDI[4]" "" "MBOQ_L" "" "")
  ("io_2_Ou"  "G3SDI[5]" "" "MBOQ_L" "" "")
  ("io_2_Ou"  "G3SDI[6]" "" "MBOQ_L" "" "")
  ("io_2_Ou"  "G3SDI[7]" "" "MBOQ_L" "" "")
  ("gnd_2_Ou" "GND" "" "" "" "")
  ("io_2_Ou"  "B3SDI[0]" "" "MB10Q_L" "" "")
  ("io_2_Ou"  "B3SDI[1]" "" "MB10Q_L" "" "")
  ("io_2_Ou"  "B3SDI[2]" "" "MB10Q_L" "" "")
  ("io_2_Ou"  "B3SDI[3]" "" "MB10Q_L" "" "")
  ("io_2_Ou"  "B3SDI[4]" "" "MB10Q_L" "" "")
  ("io_2_Ou"  "B3SDI[5]" "" "MB10Q_L" "" "")
  ("io_2_Ou"  "B3SDI[6]" "" "MB10Q_L" "" "")
  ("io_2_Ou"  "B3SDI[7]" "" "MB10Q_L" "" "")
  ("in2_2_Ou" "MB5" "" "" "" "")
  ("in2_2_Ou" "MB1" "" "" "" "")
  ("in2_2_Ou" "MB11" "" "" "" "")
  ("in2_2_Ou" "MB7" "" "" "" "")))
(epads
 (("out_2_Ou"  "R2SDI[0]" "" "" "" "")
  ("out_2_Ou"  "R2SDI[1]" "" "" "" "")
```

```
  ("out_2_Ou"  "R2SDI[2]" "" "" "" "")
  ("out_2_Ou"  "R2SDI[3]" "" "" "" "")
  ("out_2_Ou"  "R2SDI[4]" "" "" "" "")
  ("out_2_Ou"  "R2SDI[5]" "" "" "" "")
  ("out_2_Ou"  "R2SDI[6]" "" "" "" "")
  ("out_2_Ou"  "R2SDI[7]" "" "" "" "")
  ("vdd_2_Ou" "Vdd" "" "" "" "")
  ("out_2_Ou"  "G2SDI[0]" "" "" "" "")
  ("out_2_Ou"  "G2SDI[1]" "" "" "" "")
  ("out_2_Ou"  "G2SDI[2]" "" "" "" "")
  ("out_2_Ou"  "G2SDI[3]" "" "" "" "")
  ("out_2_Ou"  "G2SDI[4]" "" "" "" "")
  ("out_2_Ou"  "G2SDI[5]" "" "" "" "")
  ("out_2_Ou"  "G2SDI[6]" "" "" "" "")
  ("out_2_Ou"  "G2SDI[7]" "" "" "" "")
  ("gnd_2_Ou" "GND" "" "" "" "")
  ("out_2_Ou"  "B2SDI[0]" "" "" "" "")
  ("out_2_Ou"  "B2SDI[1]" "" "" "" "")
  ("out_2_Ou"  "B2SDI[2]" "" "" "" "")
  ("out_2_Ou"  "B2SDI[3]" "" "" "" "")
  ("out_2_Ou"  "B2SDI[4]" "" "" "" "")
  ("out_2_Ou"  "B2SDI[5]" "" "" "" "")
  ("out_2_Ou"  "B2SDI[6]" "" "" "" "")
  ("out_2_Ou"  "B2SDI[7]" "" "" "" "")
  ("in2_2_Ou" "MB6" "" "" "" "")
  ("in2_2_Ou" "MB12" "" "" "" "")))
(spads
 (("in2_2_Ou" "MB8" "" "" "" "")
  ("out_2_Ou" "ACK_L" "" "" "" "")
  ("in2_2_Ou" "R3SDII[0]" "" "" "" "")
  ("in2_2_Ou" "R3SDII[1]" "" "" "" "")
  ("in2_2_Ou" "R3SDII[2]" "" "" "" "")
  ("in2_2_Ou" "R3SDII[3]" "" "" "" "")
  ("in2_2_Ou" "R3SDII[4]" "" "" "" "")
  ("in2_2_Ou" "R3SDII[5]" "" "" "" "")
  ("in2_2_Ou" "R3SDII[6]" "" "" "" "")
  ("in2_2_Ou" "R3SDII[7]" "" "" "" "")
  ("vdd_2_Ou" "Vdd" "" "" "" "")
  ("in2_2_Ou" "G3SDII[0]" "" "" "" "")
  ("in2_2_Ou" "G3SDII[1]" "" "" "" "")
  ("in2_2_Ou" "G3SDII[2]" "" "" "" "")
  ("in2_2_Ou" "G3SDII[3]" "" "" "" "")
  ("in2_2_Ou" "G3SDII[4]" "" "" "" "")
  ("in2_2_Ou" "G3SDII[5]" "" "" "" "")
  ("in2_2_Ou" "G3SDII[6]" "" "" "" "")
  ("in2_2_Ou" "G3SDII[7]" "" "" "" "")
  ("gnd_2_Ou" "GND" "" "" "" "")
  ("in2_2_Ou" "B3SDII[0]" "" "" "" "")
  ("in2_2_Ou" "B3SDII[1]" "" "" "" "")
  ("in2_2_Ou" "B3SDII[2]" "" "" "" "")
  ("in2_2_Ou" "B3SDII[3]" "" "" "" "")
  ("in2_2_Ou" "B3SDII[4]" "" "" "" "")
  ("in2_2_Ou" "B3SDII[5]" "" "" "" "")
  ("in2_2_Ou" "B3SDII[6]" "" "" "" "")
  ("in2_2_Ou" "B3SDII[7]" "" "" "" "")
  ("in2_2_Ou" "MB9" "" "" "" "")))
```

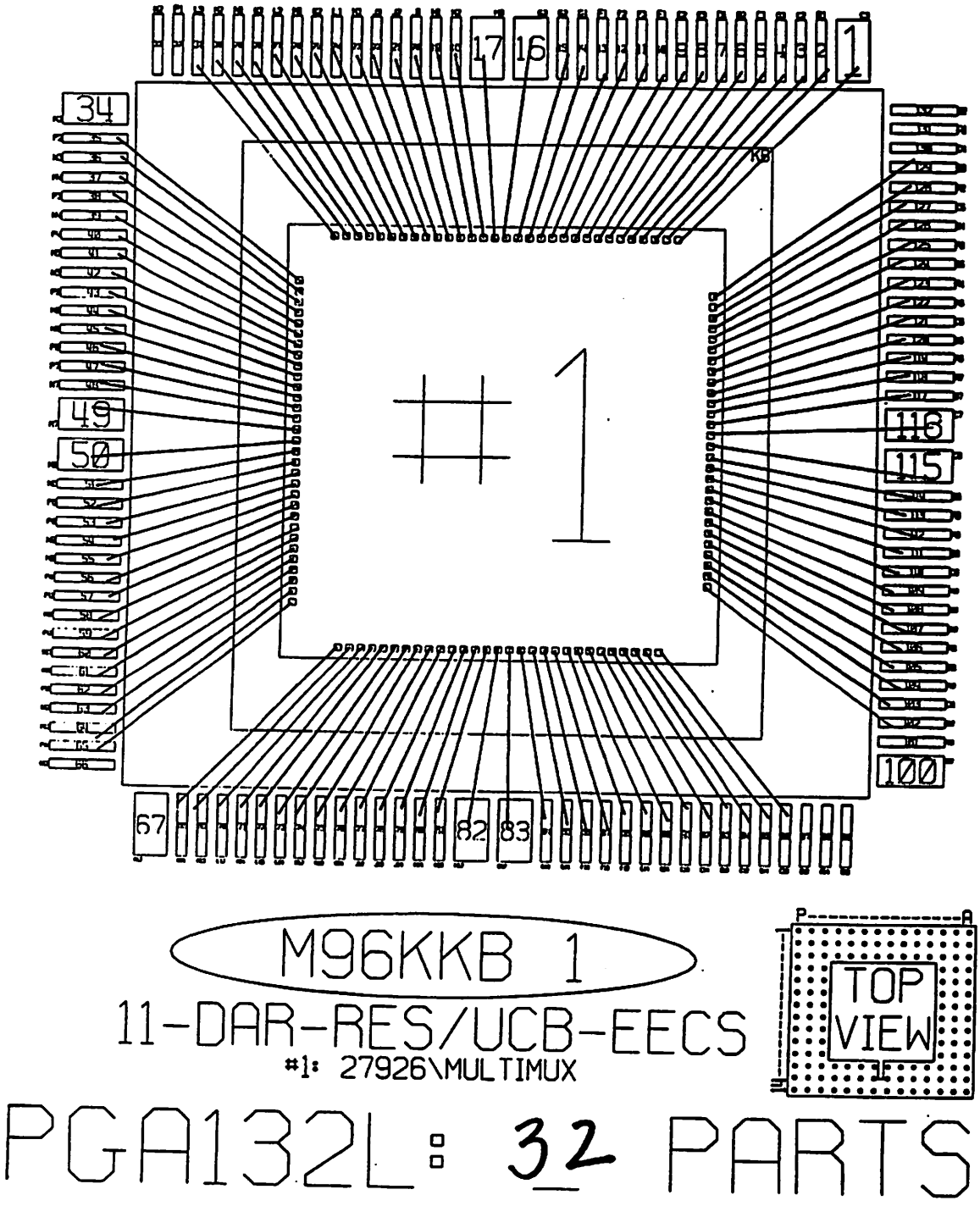


Figure A.1: Bonding Diagram of Three-channel Multiplexor.

ONE-CHANNEL_MUX.par

```
(# 8)
(npads
 (("in2_2_0u"  "P_OUT[0]" "" "" "" "")
  ("in2_2_0u" "P_OUT[1]" "" "" "" "")
  ("in2_2_0u" "P_OUT[2]" "" "" "" "")
  ("in2_2_0u"  "P_OUT[3]" "" "" "" "")
  ("in2_2_0u"  "P_OUT[4]" "" "" "" "")
  ("in2_2_0u"  "P_OUT[5]" "" "" "" "")
  ("in2_2_0u"  "P_OUT[6]" "" "" "" "")
  ("in2_2_0u"  "P_OUT[7]" "" "" "" "")
  ("gnd_2_0u" "GND" "" "" "" "")
  ("in2_2_0u" "PHASE1" "" "" "" "")
  ("in2_2_0u" "PHASE2" "" "" "" "")
  ("vdd_2_0u" "Vdd" "" "" "" "")
  ("in2_2_0u" "CS_L" "" "" "" "")))
(spads
  (("vdd_2_0u" "Vdd" "" "" "" "")
  ("out_2_0u" "P_IN[0]" "" "" "" "")
  ("out_2_0u" "P_IN[1]" "" "" "" "")
  ("out_2_0u" "P_IN[2]" "" "" "" "")
  ("out_2_0u" "P_IN[3]" "" "" "" "")
  ("out_2_0u" "P_IN[4]" "" "" "" "")
  ("out_2_0u" "P_IN[5]" "" "" "" "")
  ("out_2_0u" "P_IN[6]" "" "" "" "")
  ("out_2_0u" "P_IN[7]" "" "" "" "")
  ("gnd_2_0u" "GND" "" "" "" "")
  ("in2_2_0u" "SEL1" "" "" "" "")
  ("in2_2_0u" "SEL2" "" "" "" "")
  ("in2_2_0u" "SEL3" "" "" "" "")
  ("out_2_0u" "ACK_L" "" "" "" "")
  ("gnd_2_0u" "GND" "" "" "" "")))
(epads
 (("out_2_0u"  "FB_IN[0]" "" "" "" "")
  ("out_2_0u"  "FB_IN[1]" "" "" "" "")
  ("out_2_0u"  "FB_IN[2]" "" "" "" "")
  ("out_2_0u"  "FB_IN[3]" "" "" "" "")
  ("out_2_0u"  "FB_IN[4]" "" "" "" "")
  ("out_2_0u"  "FB_IN[5]" "" "" "" "")
  ("out_2_0u"  "FB_IN[6]" "" "" "" "")
  ("out_2_0u"  "FB_IN[7]" "" "" "" "")
  ("vdd_2_0u" "Vdd" "" "" "" "")
  ("out_2_0u"  "DA_IN[0]" "" "" "" "")
  ("out_2_0u"  "DA_IN[1]" "" "" "" "")
  ("out_2_0u"  "DA_IN[2]" "" "" "" "")
  ("out_2_0u"  "DA_IN[3]" "" "" "" "")
  ("out_2_0u"  "DA_IN[4]" "" "" "" "")
  ("out_2_0u"  "DA_IN[5]" "" "" "" "")
  ("out_2_0u"  "DA_IN[6]" "" "" "" "")
  ("out_2_0u"  "DA_IN[7]" "" "" "" "")))
(wpads
 (("in2_2_0u" "FB_OUT[0]" "" "" "" "")
  ("in2_2_0u" "FB_OUT[1]" "" "" "" "")
  ("in2_2_0u" "FB_OUT[2]" "" "" "" "")
  ("in2_2_0u" "FB_OUT[3]" "" "" "" "")
  ("in2_2_0u" "FB_OUT[4]" "" "" "" "")
  ("in2_2_0u" "FB_OUT[5]" "" "" "" "")
  ("in2_2_0u" "FB_OUT[6]" "" "" "" "")
  ("in2_2_0u" "FB_OUT[7]" "" "" "" "")
  ("vdd_2_0u" "Vdd" "" "" "" "")
  ("in2_2_0u" "AD_OUT[0]" "" "" "" "")
  ("in2_2_0u" "AD_OUT[1]" "" "" "" "")
  ("in2_2_0u" "AD_OUT[2]" "" "" "" "")
  ("in2_2_0u" "AD_OUT[3]" "" "" "" "")
  ("in2_2_0u" "AD_OUT[4]" "" "" "" "")
  ("in2_2_0u" "AD_OUT[5]" "" "" "" "")
  ("in2_2_0u" "AD_OUT[6]" "" "" "" "")
  ("in2_2_0u" "AD_OUT[7]" "" "" "" "")))
```

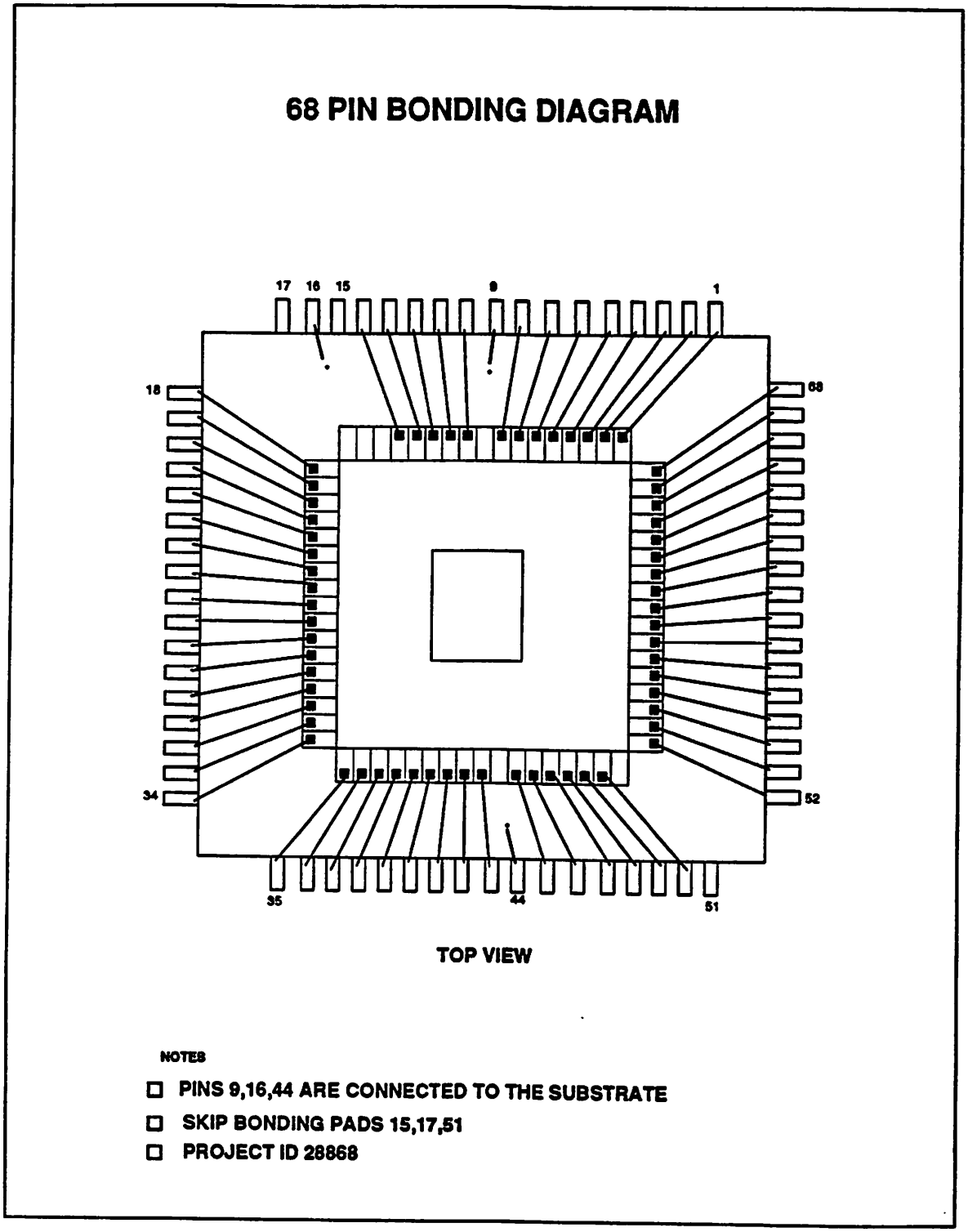


Figure A.2: Bonding Diagram For One-channel ASIC.

Appendix B

Sample Board Sdl File

board.sdl

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;                  NAME:IMAGE PROCESSING BOARD (TOP LEVEL)           ;;
;;     This module contains the connectivity information for the      ;;
;;     top level. It calls various modules (processors, vme interface, ;;
;;     bypass capacitors, and PROTO) and contains connector and       ;;
;;     and multiplexor interconnection information                    ;;
;;     Author: Anantha P.Chandrakasan                                 ;;
;;     CALLED BY: NONE                                                ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;     All the sdl files for the image board desing can be found in   ;;
;;     ~anantha/board/TOP and ~anantha/board/vme_interface            ;;
;;                                                                    ;;
;;     The leaf-cell sdl files can be found in:                       ;;
;;     ~anantha/board/cellib                                          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(parent-cell board)

(subcells
        (pip (pip_1))
        (bip (bip_1))
        (vme (vme_1))
(TCNMUX  MUXFB1 ((ROTATION "180.0") (POSITION "8.60 10.40")))
(TCNMUX  MUXFB2 ((ROTATION "180.0") (POSITION "8.60 8.50")))
(TCNMUX  MUXVP1 ((ROTATION "180.0") (POSITION "8.60 6.60")))
(TCNMUX  MUXVP2 ((ROTATION "180.0") (POSITION "8.60 4.70")))
        (OCHMUX  MUXPI ((ROTATION "90.0") (POSITION "3.50 9.10")))
        (CON60  R1 ((ROTATION "180.0") (POSITION "12.3 6.70")))
        (CON60  R2 ((ROTATION "180.0") (POSITION "12.3 7.90")))
        (CON60  G1 ((ROTATION "180.0") (POSITION "12.3 5.50")))
        (CON60  G2 ((ROTATION "180.0") (POSITION "12.3 9.10")))
        (CON60  B1 ((ROTATION "180.0") (POSITION "12.3 4.30")))
        (CON60  B2 ((ROTATION "180.0") (POSITION "12.3 10.30")))
        (CON14  RI3 ((ROTATION "180.0") (POSITION "2.95 10.30")))
        (CON14  GI3 ((ROTATION "180.0") (POSITION "1.90 10.30")))
        (CON14  BI3 ((ROTATION "180.0") (POSITION "0.85 10.30")))
        (CON14  RO3 ((ROTATION "180.0") (POSITION "6.1 10.30")))
        (CON14  GO3 ((ROTATION "180.0") (POSITION "5.05 10.30")))
        (CON14  BO3 ((ROTATION "180.0") (POSITION "4.0 10.30")))
        (CON60  SPARE ((ROTATION "180.0") (POSITION "12.3 3.10")))
        (xx244 xx244_1 ((ROTATION "180.0") (POSITION "13.80 10.60")))
        (xx244 xx244_2 ((ROTATION "180.0") (POSITION "13.80 10.00")))
        (xx244 xx244_3 ((ROTATION "180.0") (POSITION "13.80 9.40")))
        (xx244 xx244_4 ((ROTATION "180.0") (POSITION "13.80 8.80")))
        (xx244 xx244_5 ((ROTATION "180.0") (POSITION "13.80 8.20")))
        (xx244 xx244_6 ((ROTATION "180.0") (POSITION "13.80 7.60")))
        (xx244 xx244_7 ((ROTATION "180.0") (POSITION "3.60 8.80")))
        (xx244 xx244_8 ((ROTATION "180.0") (POSITION "2.30 8.30")))
        (xx244 xx244_9 ((ROTATION "180.0") (POSITION "3.40 8.30")))
        (xx244 xx244_10 ((ROTATION "180.0") (POSITION "13.80 7.00")))
        (xx244 xx244_11 ((ROTATION "180.0") (POSITION "13.80 6.40")))
        (PROTO PROTO_1 ((ROTATION "0.0") (POSITION "0.0 0.0")))
        (capacitor capacitor_1 ((ROTATION "0.0") (POSITION "0.0 0.0")))

)
(layout-generator oct2rinf)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                     POWER NETS                                   ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(net GND ((parent GND) (bip_1 GND) (pip_1 GND) (vme_1 GND) (vme_1 GNDBUS
  (MERGE 0 30)) (vme_1 AGND (MERGE 0 5)) (MUXFB1 GND (MERGE 0 3))
  (MUXFB2 GND (MERGE 0 3)) (MUXVP1 GND (MERGE 0 3)) (MUXVP2 GND
  (MERGE 0 3)) (MUXPI GND (MERGE 0 2)) (xx244_1 GND) (xx244_2 GND)
  (xx244_3 GND) (xx244_4 GND) (xx244_5 GND) (xx244_6 GND)
```

```
  (xx244_7 GND) (xx244_8 GND) (xx244_9 GND) (xx244_10 GND)
  (xx244_2 G1_L) (xx244_1 G1_L) (xx244_11 GND)
  (xx244_3 G1_L) (xx244_4 G1_L) (xx244_5 G1_L) (xx244_6 G1_L)
  (xx244_7 G1_L) (xx244_8 G1_L) (xx244_9 G1_L) (xx244_11 G2_L)
  (xx244_1 G2_L) (xx244_2 G2_L) (xx244_10 G1_L) (xx244_11 G1_L)
  (xx244_3 G2_L) (xx244_4 G2_L) (xx244_5 G2_L) (xx244_6 G2_L)
         (xx244_7 G2_L) (xx244_8 G2_L) (xx244_9 G2_L) (xx244_10 G2_L)
  (PROTO_1 GND) (RO3 B 3) (RO3 B 0) (RO3 A 5) (RO3 A 2)
  (GO3 B 3) (GO3 B 0) (GO3 A 5) (GO3 A 2)
  (BO3 B 3) (BO3 B 0) (BO3 A 5) (BO3 A 2)
  (RI3 B 3) (RI3 B 0) (RI3 A 5) (RI3 A 2)
  (GI3 B 3) (GI3 B 0) (GI3 A 5) (GI3 A 2)
;   GND WERE DONE IN RACAL FOR GRB1-3 connecors
  (BI3 B 3) (BI3 B 0) (BI3 A 5) (BI3 A 2)
         (capacitor_1 GND)
  ))

(net VCC ((parent VCC) (bip_1 VCC) (pip_1 VCC) (vme_1 VCC) (vme_1 VCCBUS
         (MERGE 0 30)) (MUXPI VCC (MERGE 0 3)) (MUXFB1 VCC (MERGE 0 3))
  (MUXFB2 VCC (MERGE 0 3)) (MUXVP1 VCC (MERGE 0 3)) (MUXVP2 VCC
  (MERGE 0 3)) (MUXFB1 SUB) (MUXFB2 SUB) (MUXVP1 SUB) (MUXVP2 SUB)
  (MUXPI SUB (MERGE 0 2)) (xx244_9 VCC) (xx244_10 VCC) (xx244_11 VCC)
  (xx244_1 VCC ) (xx244_2 VCC) (xx244_3 VCC) (xx244_4 VCC)
  (xx244_5 VCC ) (xx244_6 VCC) (xx244_7 VCC) (xx244_8 VCC)
         (capacitor_1 VCC)
  (PROTO_1 VCC)))

(net PLUS12 ((parent PLUS12) (vme_1 PLUS12 (MERGE 0 4))))
(net NEG12  ((parent NEG12) (vme_1 NEG12 (MERGE 0 4))))
(net VEE ((parent VEE) (vme_1 VEE (MERGE 0 5))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;   nets(DATA PATH)      ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; From A/D out R1,G1,and B1 connectors

(net ADINRE (NETWIDTH 4) ((MUXFB1 R1SDI 0 2) (R1 A 14)))
(net ADINRO (NETWIDTH 4) ((MUXFB1 R1SDI 1 2) (R1 B 14)))
(net ADINGE (NETWIDTH 4) ((MUXFB1 G1SDI 0 2) (G1 A 14)))
(net ADINGO (NETWIDTH 4) ((MUXFB1 G1SDI 1 2) (G1 B 14)))
(net ADINBE (NETWIDTH 4) ((MUXFB1 B1SDI 0 2) (B1 A 14)))
(net ADINBO (NETWIDTH 4) ((MUXFB1 B1SDI 1 2) (B1 B 14)))

;

(net RINT1 (NETWIDTH 8) ((MUXFB1 R3SDI 0) (MUXFB2 R3SDI 0) (MUXVP1 R1SDI 0)))
(net GINT1 (NETWIDTH 8) ((MUXFB1 G3SDI 0) (MUXFB2 G3SDI 0) (MUXVP1 G1SDI 0)))
(net BINT1 (NETWIDTH 8) ((MUXFB1 B3SDI 0) (MUXFB2 B3SDI 0) (MUXVP1 B1SDI 0)))

;

(net RINT2 (NETWIDTH 8) ((MUXFB1 R3SDII 0) (MUXFB2 R3SDII 0) (MUXVP2 R2SDI 0)))
(net GINT2 (NETWIDTH 8) ((MUXFB1 G3SDII 0) (MUXFB2 G3SDII 0) (MUXVP2 G2SDI 0)))
(net BINT2 (NETWIDTH 8) ((MUXFB1 B3SDII 0) (MUXFB2 B3SDII 0) (MUXVP2 B2SDI 0)))

; To the frame buffers R2,G2,and B2

(net RFBINE (NETWIDTH 4) ((MUXFB1 R2SDI 0 2) (xx244_1 A1 0)))
(net RFBINO (NETWIDTH 4) ((MUXFB1 R2SDI 1 2) (xx244_1 A2 0)))
(net GFBINE (NETWIDTH 4) ((MUXFB1 G2SDI 0 2) (xx244_2 A1 0)))
(net GFBINO (NETWIDTH 4) ((MUXFB1 G2SDI 1 2) (xx244_2 A2 0)))
(net BFBINE (NETWIDTH 4) ((MUXFB1 B2SDI 0 2) (xx244_3 A1 0)))
(net BFBINO (NETWIDTH 4) ((MUXFB1 B2SDI 1 2) (xx244_3 A2 0)))

(net RFBINE1 (NETWIDTH 4) ((xx244_1 Y1 0) (R2 A 14)))
(net RFBINO1 (NETWIDTH 4) ((xx244_1 Y2 0) (R2 B 14)))
(net GFBINE1 (NETWIDTH 4) ((xx244_2 Y1 0) (G2 A 14)))
(net GFBINO1 (NETWIDTH 4) ((xx244_2 Y2 0) (G2 B 14)))
(net BFBINE1 (NETWIDTH 4) ((xx244_3 Y1 0) (B2 A 14)))
```

```
(net BFBIN01 (NETWIDTH 4) ((xx244_3 Y2 0) (B2 B 14)))

;FROM the frame buffers R2, G2, and ,B2;

(net RFBOUTE (NETWIDTH 4) ((MUXFB2 R1SDI 0 2) (R2 A 22)))
(net RFBOUTO (NETWIDTH 4) ((MUXFB2 R1SDI 1 2) (R2 B 22)))
(net GFBOUTE (NETWIDTH 4) ((MUXFB2 G1SDI 0 2) (G2 A 22)))
(net GFBOUTO (NETWIDTH 4) ((MUXFB2 G1SDI 1 2) (G2 B 22)))
(net BFBOUTE (NETWIDTH 4) ((MUXFB2 B1SDI 0 2) (B2 A 22)))
(net BFBOUTO (NETWIDTH 4) ((MUXFB2 B1SDI 1 2) (B2 B 22)))

; To D/A R1,G1,andB1 connectors

(net DAINRE (NETWIDTH 4) ((MUXFB2 R2SDI 0 2) (xx244_4 A1 0)))  ;0,2,4,6
(net DAINRO (NETWIDTH 4) ((MUXFB2 R2SDI 1 2) (xx244_4 A2 0)))
(net DAINGE (NETWIDTH 4) ((MUXFB2 G2SDI 0 2) (xx244_5 A1 0)))
(net DAINGO (NETWIDTH 4) ((MUXFB2 G2SDI 1 2) (xx244_5 A2 0)))
(net DAINBE (NETWIDTH 4) ((MUXFB2 B2SDI 0 2) (xx244_6 A1 0)))
(net DAINBO (NETWIDTH 4) ((MUXFB2 B2SDI 1 2) (xx244_6 A2 0)))

(net DAINRE1 (NETWIDTH 4) ((xx244_4 Y1 0) (R1 A 22)))
(net DAINRO1 (NETWIDTH 4) ((xx244_4 Y2 0) (R1 B 22)))
(net DAINGE1 (NETWIDTH 4) ((xx244_5 Y1 0) (G1 A 22)))
(net DAINGO1 (NETWIDTH 4) ((xx244_5 Y2 0) (G1 B 22)))
(net DAINBE1 (NETWIDTH 4) ((xx244_6 Y1 0) (B1 A 22)))
(net DAINBO1 (NETWIDTH 4) ((xx244_6 Y2 0) (B1 B 22)))

; To Video port (DATA OUT)expansion connectors RO3 GO3 and BO3.

(net RVPINL (NETWIDTH 4) ((MUXVP1 R2SDI 0) (xx244_7 A1 0)))  ;0,1,2,3
(net RVPINH (NETWIDTH 4) ((MUXVP1 R2SDI 4) (xx244_7 A2 0)))  ;4,5,6,7
(net GVPINL (NETWIDTH 4) ((MUXVP1 G2SDI 0) (xx244_8 A1 0)))
(net GVPINH (NETWIDTH 4) ((MUXVP1 G2SDI 4) (xx244_8 A2 0)))
(net BVPINL (NETWIDTH 4) ((MUXVP1 B2SDI 0) (xx244_9 A1 0)))
(net BVPINH (NETWIDTH 4) ((MUXVP1 B2SDI 4) (xx244_9 A2 0)))

(net RVPIN1 (NETWIDTH 1) ((RO3 A 1)     (xx244_7 Y1 0)))     ;0
(net RVPIN2 (NETWIDTH 2) ((RO3 B 1)     (xx244_7 Y1 1)))     ;1,2
(net RVPIN3 (NETWIDTH 1) ((RO3 A 3)     (xx244_7 Y1 3)))     ;3
(net RVPIN4 (NETWIDTH 1) ((RO3 A 4)     (xx244_7 Y2 0)))     ;4
(net RVPIN5 (NETWIDTH 2) ((RO3 B 4)     (xx244_7 Y2 1)))     ;5,6
(net RVPIN6 (NETWIDTH 1) ((RO3 A 6)     (xx244_7 Y2 3)))     ;7

(net GVPIN1 (NETWIDTH 1) ((GO3 A 1)     (xx244_8 Y1 0)))     ;0
(net GVPIN2 (NETWIDTH 2) ((GO3 B 1)     (xx244_8 Y1 1)))     ;1,2
(net GVPIN3 (NETWIDTH 1) ((GO3 A 3)     (xx244_8 Y1 3)))     ;3
(net GVPIN4 (NETWIDTH 1) ((GO3 A 4)     (xx244_8 Y2 0)))     ;4
(net GVPIN5 (NETWIDTH 2) ((GO3 B 4)     (xx244_8 Y2 1)))     ;5,6
(net GVPIN6 (NETWIDTH 1) ((GO3 A 6)     (xx244_8 Y2 3)))     ;7

(net BVPIN1 (NETWIDTH 1) ((BO3 A 1)     (xx244_9 Y1 0)))     ;0
(net BVPIN2 (NETWIDTH 2) ((BO3 B 1)     (xx244_9 Y1 1)))     ;1,2
(net BVPIN3 (NETWIDTH 1) ((BO3 A 3)     (xx244_9 Y1 3)))     ;3
(net BVPIN4 (NETWIDTH 1) ((BO3 A 4)     (xx244_9 Y2 0)))     ;4
(net BVPIN5 (NETWIDTH 2) ((BO3 B 4)     (xx244_9 Y2 1)))     ;5,6
(net BVPIN6 (NETWIDTH 1) ((BO3 A 6)     (xx244_9 Y2 3)))     ;7

; From Video port to RI3 GI3 BI3

(net RVPOUT1 (NETWIDTH 1) ((RI3 A 1)     (MUXVP2 R1SDI 0)))     ;0
(net RVPOUT2 (NETWIDTH 2) ((RI3 B 1)     (MUXVP2 R1SDI 1)))     ;1,2
(net RVPOUT3 (NETWIDTH 2) ((RI3 A 3)     (MUXVP2 R1SDI 3)))     ;3,4
(net RVPOUT4 (NETWIDTH 2) ((RI3 B 4)     (MUXVP2 R1SDI 5)))     ;5,6
(net RVPOUT5 (NETWIDTH 1) ((RI3 A 6)     (MUXVP2 R1SDI 7)))     ;7

(net GVPOUT1 (NETWIDTH 1) ((GI3 A 1)     (MUXVP2 G1SDI 0)))     ;0
(net GVPOUT2 (NETWIDTH 2) ((GI3 B 1)     (MUXVP2 G1SDI 1)))     ;1,2
(net GVPOUT3 (NETWIDTH 2) ((GI3 A 3)     (MUXVP2 G1SDI 3)))     ;3,4
(net GVPOUT4 (NETWIDTH 2) ((GI3 B 4)     (MUXVP2 G1SDI 5)))     ;5,6
(net GVPOUT5 (NETWIDTH 1) ((GI3 A 6)     (MUXVP2 G1SDI 7)))     ;7
```

```
(net BVPOUT1 (NETWIDTH 1) ((BI3 A 1)     (MUXVP2 B1SDI 0)))     ;0
(net BVPOUT2 (NETWIDTH 2) ((BI3 B 1)     (MUXVP2 B1SDI 1)))     ;1,2
(net BVPOUT3 (NETWIDTH 2) ((BI3 A 3)     (MUXVP2 B1SDI 3)))     ;3,4
(net BVPOUT4 (NETWIDTH 2) ((BI3 B 4)     (MUXVP2 B1SDI 5)))     ;5,6
(net BVPOUT5 (NETWIDTH 1) ((BI3 A 6)     (MUXVP2 B1SDI 7)))     ;7

; nets to pip

(net GPIPIN (NETWIDTH 8) ((MUXVP1 G3SDI 0) (MUXPI AD_OUT 0) (MUXVP2 G3SDI 0)))
(net GPIPIN2 (NETWIDTH 8) ((MUXPI P_IN 0) (pip_1 DI 0)))
(net GPIPOUT (NETWIDTH 8) ((MUXPI P_OUT 0) (pip_1 DO 0)))
(net GBIPIN (NETWIDTH 8) ((MUXPI FB_IN 0) (bip_1 DI 0)))
(net GBIPOUT (NETWIDTH 8) ((MUXPI FB_OUT 0) (bip_1 DOUT 0)))
(net DA_ING (NETWIDTH 8) ((MUXPI DA_IN 0) (MUXVP1 G3SDII 0) (MUXVP2 G3SDII 0)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;   SIGNALS FROM THE R1-> R2 G1->G2 B1->B2                                ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(net R1D1 (NETWIDTH 4) ((R1 A 18) (R2 A 18)))
(net R1D2 (NETWIDTH 4) ((R1 B 18) (R2 B 18)))
(net G1D1 (NETWIDTH 4) ((G1 A 18) (G2 A 18)))
(net G1D2 (NETWIDTH 4) ((G1 B 18) (G2 B 18)))
(net B1D1 (NETWIDTH 4) ((B1 A 18) (B2 A 18)))
(net B1D2 (NETWIDTH 4) ((B1 B 18) (B2 B 18)))

(net R1D3 (NETWIDTH 4) ((R1 A 26) (R2 A 26)))
(net R1D4 (NETWIDTH 4) ((R1 B 26) (R2 B 26)))
(net G1D3 (NETWIDTH 4) ((G1 A 26) (G2 A 26)))
(net G1D4 (NETWIDTH 4) ((G1 B 26) (G2 B 26)))
(net B1D3 (NETWIDTH 4) ((B1 A 26) (B2 A 26)))
(net B1D4 (NETWIDTH 4) ((B1 B 26) (B2 B 26)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;   SIGNALS FROM THE R1-> R2 G1->G2 B1->B2 and to the board       ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(net XTAL1     ((R1 B 0) (R2 B 0)))
(net SYSCLK1   ((R1 B 1) (R2 B 1)))
(net HRESET_L1 ((R1 B 2) (R2 B 2)))
(net VRESET_L1 ((R1 B 3) (R2 B 3)))
(net DOTCLK_L1 ((R1 B 4) (R2 B 4)))
(net EBLANK_L1 ((R1 B 6) (R2 B 6)))
(net VBLANK_L1 ((R1 B 7) (R2 B 7)))
(net VSYNC_L1  ((R1 B 8) (R2 B 8)))
(net FIELD1    ((R1 B 9) (R2 B 9)))
(net HREF_L1   ((R1 B 10) (R2 B 10)))

(net HBLANK_L1 ((R1 A 7) (R2 A 7)))
(net HSYNC_L1 ((R1 A 8) (R2 A 8)))
(net CSYNC_L1 ((R1 A 9) (R2 A 9)))
(net SBLANK_L1 ((R1 A 10) (R2 A 10)))
(net VGATE_L1 ((R1 A 11) (R2 A 11)))

(net XTAL2    ((G1 B 0) (G2 B 0)))
(net SYSCLK2  ((G1 B 1) (G2 B 1)))
(net HRESET_L2 ((G1 B 2) (G2 B 2)))
(net VRESET_L2 ((G1 B 3) (G2 B 3)))
(net DOTCLK_L2 ((G1 B 4) (G2 B 4)))
(net EBLANK_L2 ((G1 B 6) (G2 B 6)))
(net VBLANK_L2 ((G1 B 7) (G2 B 7)))
(net VSYNC_L2  ((G1 B 8) (G2 B 8)))
(net FIELD2    ((G1 B 9) (G2 B 9)))
(net HREF_L2   ((G1 B 10) (G2 B 10)))

(net HBLANK_L2 ((G1 A 7) (G2 A 7)))
(net HSYNC_L2  ((G1 A 8) (G2 A 8)))
(net CSYNC_L2 ((G1 A 9) (G2 A 9)))
(net SBLANK_L2 ((G1 A 10) (G2 A 10)))
```

```
(net VGATE_L2 ((G1 A 11) (G2 A 11)))

(net XTAL3     ((B1 B 0) (B2 B 0)))
(net SYSCLK3   ((B1 B 1) (B2 B 1)))
(net HRESET_L3 ((B1 B 2) (B2 B 2)))
(net VRESET_L3 ((B1 B 3) (B2 B 3)))
(net DOTCLK_L3 ((B1 B 4) (B2 B 4) (xx244_10 A1 3) (xx244_10 A2 (MERGE 0 1))))
(net EBLANK_L3 ((B1 B 6) (B2 B 6)))
(net VBLANK_L3 ((B1 B 7) (B2 B 7) (xx244_10 A1 1) (xx244_11 A2 2)))
(net VSYNC_L3  ((B1 B 8) (B2 B 8)))
(net FIELD3    ((B1 B 9) (B2 B 9) (xx244_10 A1 0)))
(net HREF_L4   ((B1 B 10) (B2 B 10)))

(net HBLANK_L3 ((B1 A 7) (B2 A 7) (xx244_10 A1 2) (xx244_11 A2 3)))
(net HSYNC_L3 ((B1 A 8) (B2 A 8) (xx244_10 A2 2)))
(net CSYNC_L3 ((B1 A 9) (B2 A 9)))
(net SBLANK_3 ((B1 A 10) (B2 A 10) (xx244_10 A2 3)))
(net VGATE_3  ((B1 A 11) (B2 A 11)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;      SIGNALS TO PIP AND BIP and G3                              ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(net BDOT1_L ((xx244_10 Y1 3) (G03 A 0)))    ; was B03 A 0
(net BDOT2_L ((xx244_10 Y2 0) (pip_1 DOTCLOCK1_L)))
(net BDOT3_L ((xx244_10 Y2 1) (pip_1 DOTCLOCK2_L)))
(net BFIELD ((xx244_10 Y1 0) (pip_1 FIELD) (B03 B 6)))
(net BVBLANK_L ((xx244_10 Y1 1) (pip_1 VBLANK_L)))
(net BHBLANK_L ((xx244_10 Y1 2) (pip_1 HBLANK_L)))
(net BVBLANK2_L ((xx244_11 Y2 2) (B03 A 0)))
(net BHBLANK2_L ((xx244_11 Y2 3) (B03 A 0))) ; was G03
(net BSBLANK_L ((xx244_10 Y2 3) (B03 B 6)))  ; was G03
(net BHSYNC_L ((xx244_10 Y2 2) (pip_1 HSYNC_L)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;   CLOCKS(FOR MUX CHIPS)              ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(net DOTM1 ((pip_1 DCLKM) (xx244_11 A1 (MERGE 0 3))
                                   (xx244_11 A2 0)))
(net DOTM2 ((pip_1 DCLKM_L) (xx244_11 A2 1)))
(net MPHI1 ((xx244_11 Y1 0) (MUXFB1 BDOTCLOCK)))
(net MPHI2 ((xx244_11 Y1 1) (MUXFB2 BDOTCLOCK)))
(net MPHI3 ((xx244_11 Y1 2) (MUXVP1 BDOTCLOCK)))
(net MPHI4 ((xx244_11 Y1 3) (MUXVP2 BDOTCLOCK)))
; changed on the 2nd dec. PHASE1 & 2 are the same-- change reflected
; in the layout.
(net MPHI5 ((xx244_11 Y2 0) (MUXPI PHASE1) (MUXPI PHASE2)))
;(net MPHI5_L ((xx244_11 Y2 1) (MUXPI PHASE2)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;   CONTROL SIGNALS(To MUX CHIPS)    ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;for  multiplexors

(net D0 ((vme_1 D 0) (MUXFB1 MB4) (MUXFB2 MB4) (MUXVP1 MB4) (MUXVP2 MB4)
 (MUXPI SEL1) (pip_1 BDATA 0) (pip_1 HDATA 0)))
(net D1 ((vme_1 D 1) (MUXFB1 MB0) (MUXFB2 MB0) (MUXVP1 MB0) (MUXVP2 MB0)
 (MUXPI SEL2) (pip_1 BDATA 1) (pip_1 HDATA 1)))
(net D2 ((vme_1 D 2) (MUXFB1 MB10) (MUXFB2 MB10) (MUXVP1 MB10) (MUXVP2 MB10)
 (MUXPI SEL3) (pip_1 BDATA 2) (pip_1 HDATA 2)))
(net D3 ((vme_1 D 3) (MUXFB1 MB5) (MUXFB2 MB5) (MUXVP1 MB5) (MUXVP2 MB5)
 (pip_1 BDATA 3) (pip_1 HDATA 3)))
(net D4 ((vme_1 D 4) (MUXFB1 MB1) (MUXFB2 MB1) (MUXVP1 MB1) (MUXVP2 MB1)
 (pip_1 BDATA 4) (pip_1 HDATA 4)))
(net D5 ((vme_1 D 5) (MUXFB1 MB11) (MUXFB2 MB11) (MUXVP1 MB11) (MUXVP2 MB11)
 (pip_1 BDATA 5) (pip_1 HDATA 5)))
(net D6 ((vme_1 D 6) (MUXFB1 MB6) (MUXFB2 MB6) (MUXVP1 MB6) (MUXVP2 MB6)
 (pip_1 BDATA 6) (pip_1 HDATA 6)))
```

```
(net D7 ((vme_1 D 7) (MUXFB1 MB7) (MUXFB2 MB7) (MUXVP1 MB7) (MUXVP2 MB7)
 (pip_1 BDATA 7) (pip_1 HDATA 7)))
(net D8 ((vme_1 D 8) (MUXFB1 MB12) (MUXFB2 MB12) (MUXVP1 MB12) (MUXVP2 MB12)))
(net D9 ((vme_1 D 9) (MUXFB1 MB13) (MUXFB2 MB13) (MUXVP1 MB13) (MUXVP2 MB13)))
(net D10 ((vme_1 D 10) (MUXFB1 MB8) (MUXFB2 MB8) (MUXVP1 MB8) (MUXVP2 MB8)))
(net D11 ((vme_1 D 11) (MUXFB1 MB9) (MUXFB2 MB9) (MUXVP1 MB9) (MUXVP2 MB9)))

(net CS_L1 ((vme_1 MSEL_L 0) (MUXFB1 CS_L)))
(net CS_L2 ((vme_1 MSEL_L 1) (MUXFB2 CS_L)))
(net CS_L3 ((vme_1 MSEL_L 2) (MUXVP1 CS_L)))
(net CS_L4 ((vme_1 MSEL_L 3) (MUXVP2 CS_L)))
(net CS_L5 ((vme_1 MSEL_L 4 ) (MUXPI CS_L)))

(net ACK_L1 ((vme_1 ACK_L 0) (MUXFB1 ACK_L)))
(net ACK_L2 ((vme_1 ACK_L 1) (MUXFB2 ACK_L)))
(net ACK_L3 ((vme_1 ACK_L 2) (MUXVP1 ACK_L)))
(net ACK_L4 ((vme_1 ACK_L 3) (MUXVP2 ACK_L)))
(net ACK_L5 ((vme_1 ACK_L 4) (MUXPI ACK_L)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;   CONTROL SIGNALS(To pip)    ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(net WSEL (NETWIDTH 7) ((vme_1 WSEL 0) (pip_1 WSEL 1)))
(net RSEL_L (NETWIDTH 4) ((vme_1 RSEL_L) (pip_1 SEL 0)))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;   FROM clock generation pip to bip                                        ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(net INLATCHT   ((bip_1 INLATCHB)   (pip_1 INLATCHB)))
(net INLATCH_LT ((bip_1 OUTLATCHB) (pip_1 OUTLATCHB)))
(net PHI1HT     ((bip_1 HPHI1) (pip_1 HPHI1)))
(net PHI2HT     ((bip_1 HPHI2) (pip_1 HPHI2)))
(net PHI1CT     ((bip_1 CPHI1) (pip_1 CPHI1)))
(net PHI2CT     ((bip_1 CPHI2) (pip_1 CPHI2)))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;   FROM THE PLD IN pip to bip                                              ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(net VBLANK ((pip_1 VBLANK) (bip_1 VBLANK)))
(net HSYNC ((pip_1 HSYNC) (bip_1 HSYNC)))
(net HBLANK ((pip_1 HBLANK) (bip_1 HBLANK)))
(net HBLANKB_L ((pip_1 HBLANKB_L) (bip_1 HBLANKB_L)))
(net LINESTART ((pip_1 LINESTART) (bip_1 LINESTART)))
(net FIELD_L ((pip_1 FIELD_L) (bip_1 FIELD_L)))

(end-sdl)
```

Appendix C

Connector Pin Mapping

R1 CONNECTOR

R1(FROM AND TO A/D)

R-C	SIGNAL NAME	R-C	SIGNAL NAME
B29	R1SDO[15]	A29	R1SDO[14]
B28	R1SDO[13]	A28	R1SDO[12]
B27	R1SDO[11]	A27	R1SDO[10]
B26	R1SDO[9]	A26	R1SDO[8]
B25	R1SDO[7]	A25	R1SDO[6]
B24	R1SDO[5]	A24	R1SDO[4]
B23	R1SDO[3]	A23	R1SDO[2]
B22	R1SDO[1]	A22	R1SDO[0]
B21	R1SDI[15]	A21	R1SDI[14]
B20	R1SDI[13]	A20	R1SDI[12]
B19	R1SDI[11]	A19	R1SDI[10]
B18	R1SDI[9]	A18	R1SDI[8]
B17	R1SDI[7]	A17	R1SDI[6]
B16	R1SDI[5]	A16	R1SDI[4]
B15	R1SDI[3]	A15	R1SDI[2]
B14	R1SDI[1]	A14	R1SDI[0]
B13	-RESERVED-	A13	-RESERVED-
B12	-RESERVED-	A12	-RESERVED-
B11	-RESERVED-	A11	V.GATE_L
B10	H.REF_L	A10	S.BLANK_L
B9	FIELD(O/1)	A9	C.SYNC_L
B8	V.SYNC_L	A8	HSYNC_L
B7	V.BLANK_L	A7	HBLANK_L
B6	E.BLANK_L	A6	GND
B5	GND	A5	GND
B4	DOT_CLOCK_L	A4	GND
B3	V.RESET_L	A3	GND
B2	H.RESET_L	A2	GND
B1	SYSTEM CLOCK	A1	GND
B0	XTAL CLOCK	A0	GND

NOTES:

1. The pin assignments are according to Racal-Redac DB.
   B0=PIN1 : A29 PIN60;
2. R1SDI[0]-[7] are connected to R1SDI[8]-[15] respectivly.
3. R1SDO[0]-[7] are connected to R1SDO[8]-[15] respectivly.
4. R1SDI bus is from the A/D and R1SDO is to the A/D.

G1 CONNECTOR

G1(FROM AND TO A/D)

R-C	SIGNAL NAME	R-C	SIGNAL NAME
B29	G1SDO[15]	A29	G1SDO[14]
B28	G1SDO[13]	A28	G1SDO[12]
B27	G1SDO[11]	A27	G1SDO[10]
B26	G1SDO[9]	A26	G1SDO[8]
B25	G1SDO[7]	A25	G1SDO[6]
B24	G1SDO[5]	A24	G1SDO[4]
B23	G1SDO[3]	A23	G1SDO[2]
B22	G1SDO[1]	A22	G1SDO[0]
B21	G1SDI[15]	A21	G1SDI[14]
B20	G1SDI[13]	A20	G1SDI[12]
B19	G1SDI[11]	A19	G1SDI[10]
B18	G1SDI[9]	A18	G1SDI[8]
B17	G1SDI[7]	A17	G1SDI[6]
B16	G1SDI[5]	A16	G1SDI[4]
B15	G1SDI[3]	A15	G1SDI[2]
B14	G1SDI[1]	A14	G1SDI[0]
B13	-RESERVED-	A13	-RESERVED-
B12	-RESERVED-	A12	-RESERVED-
B11	-RESERVED-	A11	V.GATE_L
B10	H.REF_L	A10	S.BLANK_L
B9	FIELD(0/1)	A9	C.SYNC_L
B8	V.SYNC_L	A8	HSYNC_L
B7	V.BLANK_L	A7	HBLANK_L
B6	E.BLANK_L	A6	GND
B5	GND	A5	GND
B4	DOT_CLOCK_L	A4	GND
B3	V.RESET_L	A3	GND
B2	H.RESET_L	A2	GND
B1	SYSTEM CLOCK	A1	GND
B0	XTAL CLOCK	A0	GND

NOTES:

1. The pin assignments are according to Racal-Redac DB.
   B0=PIN1 : A29 PIN60;
2. G1SDI[0]-[7] are connected to G1SDI[8]-[15] respectivly.
3. G1SDO[0]-[7] are connected to G1SDO[8]-[15] respectivly.
4. G1SDI bus is from the A/D and G1SDO is to the A/D.

B1 CONNECTOR

B1(FROM AND TO A/D)

R-C	SIGNAL NAME	R-C	SIGNAL NAME
B29	B1SDO[15]	A29	B1SDO[14]
B28	B1SDO[13]	A28	B1SDO[12]
B27	B1SDO[11]	A27	B1SDO[10]
B26	B1SDO[9]	A26	B1SDO[8]
B25	B1SDO[7]	A25	B1SDO[6]
B24	B1SDO[5]	A24	B1SDO[4]
B23	B1SDO[3]	A23	B1SDO[2]
B22	B1SDO[1]	A22	B1SDO[0]
B21	B1SDI[15]	A21	B1SDI[14]
B20	B1SDI[13]	A20	B1SDI[12]
B19	B1SDI[11]	A19	B1SDI[10]
B18	B1SDI[9]	A18	B1SDI[8]
B17	B1SDI[7]	A17	B1SDI[6]
B16	B1SDI[5]	A16	B1SDI[4]
B15	B1SDI[3]	A15	B1SDI[2]
B14	B1SDI[1]	A14	B1SDI[0]
B13	-RESERVED-	A13	-RESERVED-
B12	-RESERVED-	A12	-RESERVED-
B11	-RESERVED-	A11	V.GATE_L
B10	H.REF_L	A10	S.BLANK_L
B9	FIELD(0/1)	A9	C.SYNC_L
B8	V.SYNC_L	A8	HSYNC_L
B7	V.BLANK_L	A7	HBLANK_L
B6	E.BLANK_L	A6	GND
B5	GND	A5	GND
B4	DOT_CLOCK_L	A4	GND
B3	V.RESET_L	A3	GND
B2	H.RESET_L	A2	GND
B1	SYSTEM CLOCK	A1	GND
B0	XTAL CLOCK	A0	GND

NOTES:

1. The pin assignments are according to Racal-Redac DB.
   B0=PIN1 : A29 PIN60;
2. B1SDI[0]-[7] are connected to B1SDI[8]-[15] respectivly.
3. B1SDO[0]-[7] are connected to B1SDO[8]-[15] respectivly.
4. B1SDI bus is from the A/D and B1SDO is to the A/D.

R2 CONNECTOR

R2(FROM AND TO FRAME BUFFER)

R-C	SIGNAL NAME	R-C	SIGNAL NAME
B29	R2SDO[15]	A29	R2SDO[14]
B28	R2SDO[13]	A28	R2SDO[12]
B27	R2SDO[11]	A27	R2SDO[10]
B26	R2SDO[9]	A26	R2SDO[8]
B25	R2SDO[7]	A25	R2SDO[6]
B24	R2SDO[5]	A24	R2SDO[4]
B23	R2SDO[3]	A23	R2SDO[2]
B22	R2SDO[1]	A22	R2SDO[0]
B21	R2SDI[15]	A21	R2SDI[14]
B20	R2SDI[13]	A20	R2SDI[12]
B19	R2SDI[11]	A19	R2SDI[10]
B18	R2SDI[9]	A18	R2SDI[8]
B17	R2SDI[7]	A17	R2SDI[6]
B16	R2SDI[5]	A16	R2SDI[4]
B15	R2SDI[3]	A15	R2SDI[2]
B14	R2SDI[1]	A14	R2SDI[0]
B13	-RESERVED-	A13	-RESERVED-
B12	-RESERVED-	A12	-RESERVED-
B11	-RESERVED-	A11	V.GATE_L
B10	H.REF_L	A10	S.BLANK_L
B9	FIELD(0/1)	A9	C.SYNC_L
B8	V.SYNC_L	A8	HSYNC_L
B7	V.BLANK_L	A7	HBLANK_L
B6	E.BLANK_L	A6	GND
B5	GND	A5	GND
B4	DOT_CLOCK_L	A4	GND
B3	V.RESET_L	A3	GND
B2	H.RESET_L	A2	GND
B1	SYSTEM CLOCK	A1	GND
B0	XTAL CLOCK	A0	GND

NOTES:

1. The pin assignments are according to Racal-Redac DB.
   B0=PIN1 : A29 PIN60;
2. R2SDI[0]-[7] are connected to R2SDI[8]-[15] respectivly.
3. R2SDO[0]-[7] are connected to R2SDO[8]-[15] respectivly.
4. R2SDO bus is from the FB and R2SDI is to the FB.

G2 CONNECTOR

G2(FROM AND TO FRAME BUFFERS)

R-C	SIGNAL NAME	R-C	SIGNAL NAME
B29	G2SDO[15]	A29	G2SDO[14]
B28	G2SDO[13]	A28	G2SDO[12]
B27	G2SDO[11]	A27	G2SDO[10]
B26	G2SDO[9]	A26	G2SDO[8]
B25	G2SDO[7]	A25	G2SDO[6]
B24	G2SDO[5]	A24	G2SDO[4]
B23	G2SDO[3]	A23	G2SDO[2]
B22	G2SDO[1]	A22	G2SDO[0]
B21	G2SDI[15]	A21	G2SDI[14]
B20	G2SDI[13]	A20	G2SDI[12]
B19	G2SDI[11]	A19	G2SDI[10]
B18	G2SDI[9]	A18	G2SDI[8]
B17	G2SDI[7]	A17	G2SDI[6]
B16	G2SDI[5]	A16	G2SDI[4]
B15	G2SDI[3]	A15	G2SDI[2]
B14	G2SDI[1]	A14	G2SDI[0]
B13	-RESERVED-	A13	-RESERVED-
B12	-RESERVED-	A12	-RESERVED-
B11	-RESERVED-	A11	V.GATE_L
B10	H.REF_L	A10	S.BLANK_L
B9	FIELD(0/1)	A9	C.SYNC_L
B8	V.SYNC_L	A8	HSYNC_L
B7	V.BLANK_L	A7	HBLANK_L
B6	H.BLANK_L	A6	GND
B5	GND	A5	GND
B4	DOT_CLOCK_L	A4	GND
B3	V.RESET_L	A3	GND
B2	H.RESET_L	A2	GND
B1	SYSTEM CLOCK	A1	GND
B0	XTAL CLOCK	A0	GND

NOTES:

1. The pin assignments are according to Racal-Redac DB.
   B0=PIN1 : A29 PIN60;
2. G2SDI[0]-[7] are connected to G2SDI[8]-[15] respectivly.
3. G2SDO[0]-[7] are connected to G2SDO[8]-[15] respectivly.
4. G2SDO bus is from the FB and G2SDI is to the FB.

B2 CONNECTOR

B2(FROM AND TO FRAME BUFFER)

R-C	SIGNAL NAME	R-C	SIGNAL NAME
B29	B2SDO[15]	A29	B2SDO[14]
B28	B2SDO[13]	A28	B2SDO[12]
B27	B2SDO[11]	A27	B2SDO[10]
B26	B2SDO[9]	A26	B2SDO[8]
B25	B2SDO[7]	A25	B2SDO[6]
B24	B2SDO[5]	A24	B2SDO[4]
B23	B2SDO[3]	A23	B2SDO[2]
B22	B2SDO[1]	A22	B2SDO[0]
B21	B2SDI[15]	A21	B2SDI[14]
B20	B2SDI[13]	A20	B2SDI[12]
B29	B2SDI[11]	A19	B2SDI[10]
B28	B2SDI[9]	A18	B2SDI[8]
B27	B2SDI[7]	A17	B2SDI[6]
B26	B2SDI[5]	A16	B2SDI[4]
B25	B2SDI[3]	A15	B2SDI[2]
B24	B2SDI[1]	A14	B2SDI[0]
B23	-RESERVED-	A13	-RESERVED-
B22	-RESERVED-	A12	-RESERVED-
B21	-RESERVED-	A11	V.GATE_L
B20	H.REF_L	A10	S.BLANK_L
B9	FIELD(O/1)	A9	C.SYNC_L
B8	V.SYNC_L	A8	HSYNC_L
B7	V.BLANK_L	A7	HBLANK_L
B6	H.BLANK_L	A6	GND
B5	GND	A5	GND
B4	DOT_CLOCK_L	A4	GND
B3	V.RESET_L	A3	GND
B2	H.RESET_L	A2	GND
B2	SYSTEM CLOCK	A1	GND
B0	XTAL CLOCK	A0	GND

NOTES:

1. The pin assignments are according to Racal-Redac DB.
   B0=PIN1 : A29 PIN60;
2. B2SDI[0]-[7] are connected to B2SDI[8]-[15] respectivly.
3. B2SDO[0]-[7] are connected to B2SDO[8]-[15] respectivly.
4. B2SDO bus is from the FB and B2SDI is to the FB.

R3

RI3 (From Video port)

R-C	SIGNAL NAME	R-C	SIGNAL NAME
A6	RVPOUT[7]	B6	NC
A5	GND	B5	RVPOUT[6]
A4	RVPOUT[4]	B4	RVPOUT[5]
A3	RVPOUT[3]	B3	GND
A2	GND	B2	RVPOUT[2]
A1	RVPOUT[0]	B1	RVPOUT[1]
A0	NC	B0	GND

NOTES:

1. The pin assignments are according to Racal-Redac DB.
   A0=PIN1 : B6 PIN14;
2. This connector is compatible with the data-cube format.

RO3 (To Video port)

R-C	SIGNAL NAME	R-C	SIGNAL NAME
A6	RVPIN[7]	B6	BFIELD
A5	GND	B5	RVPIN[6]
A4	RVPIN[4]	B4	RVPIN[5]
A3	RVPIN[3]	B3	GND
A2	GND	B2	RVPIN[2]
A1	RVPIN[0]	B1	RVPIN[1]
A0	BVLANK_L	B0	GND

NOTES:

1. The pin assignments are according to Racal-Redac DB.
   A0=PIN1 : B6 PIN14;
2. This connector is compatible with the data-cube format.

G3

GI3 (From Video port)

R-C	SIGNAL NAME	R-C	SIGNAL NAME
A6	GVPOUT[7]	B6	NC
A5	GND	B5	GVPOUT[6]
A4	GVPOUT[4]	B4	GVPOUT[5]
A3	GVPOUT[3]	B3	GND
A2	GND	B2	GVPOUT[2]
A1	GVPOUT[0]	B1	GVPOUT[1]
A0	NC	B0	GND

NOTES:

1. The pin assignments are according to Racal-Redac DB.
   A0=PIN1 : B6 PIN14;
2. This connector is compatible with the data-cube format.

GO3 (To Video port)

R-C	SIGNAL NAME	R-C	SIGNAL NAME
A6	GVPIN[7]	B6	HBLANK_L
A5	GND	B5	GVPIN[6]
A4	GVPIN[4]	B4	GVPIN[5]
A3	GVPIN[3]	B3	GND
A2	GND	B2	GVPIN[2]
A1	GVPIN[0]	B1	GVPIN[1]
A0	BDOTCLK_L	B0	GND

NOTES:

1. The pin assignments are according to Racal-Redac DB.
   A0=PIN1 : B6 PIN14;
2. This connector is compatible with the data-cube format.

B3

BI3 (From Video port)

R-C	SIGNAL NAME	R-C	SIGNAL NAME
A6	BVPOUT[7]	B6	NC
A5	GND	B5	BVPOUT[6]
A4	BVPOUT[4]	B4	BVPOUT[5]
A3	BVPOUT[3]	B3	GND
A2	GND	B2	BVPOUT[2]
A1	BVPOUT[0]	B1	BVPOUT[1]
A0	NC	B0	GND

NOTES:

1. The pin assignments are according to Racal-Redac DB.
   A0=PIN1 : B6 PIN14;
2. This connector is compatible with the data-cube format.

BO3 (To Video port)

R-C	SIGNAL NAME	R-C	SIGNAL NAME
A6	BVPIN[7]	B6	BHBLANK_L
A5	GND	B5	BVPIN[6]
A4	BVPIN[4]	B4	BVPIN[5]
A3	BVPIN[3]	B3	GND
A2	GND	B2	BVPIN[2]
A1	BVPIN[0]	B1	BVPIN[1]
A0	BDOTCLK_L	B0	GND

NOTES:

1. The pin assignments are according to Racal-Redac DB.
   A0=PIN1 : B6 PIN14;
2. This connector is compatible with the data-cube format.

Appendix D

PLD PINOUTS

decoder.rpt

```
                EP610
              - - - - -
         Gnd -|1      24|- Vcc
     BLWORDL -|2      23|- BA14
     WENABLE -|3      22|- AM0
     RENABLE -|4      21|- AM1
     MENABLE -|5      20|- AM2
        BSEL -|6      19|- AM3
        BA13 -|7      18|- AM4
        BA12 -|8      17|- AM5
        BA11 -|9      16|- BASL
        BA10 -|10     15|- BA9
      BIACKL -|11     14|- BA15
         GND -|12     13|- Gnd
              - - - - -
```

hshake.rpt

```
                EP610
              - - - - -
         Gnd -|1      24|- Vcc
      VMEWRI -|2      23|- DEVWRI
         SDC -|3      22|- Gnd
       MREQO -|4      21|- Gnd
      MREQLO -|5      20|- Gnd
       DACKO -|6      19|- Gnd
         CSLO -|7     18|- Gnd
       BREQO -|8      17|- Gnd
        CINI -|9      16|- Gnd
       BACKI -|10     15|- Gnd
       MACKI -|11     14|- DREQI
         GND -|12     13|- Gnd
              - - - - -
```

logic.rpt

```
                R  R  R                    R  R  R  R
                E  E  E                    E  E  E  E
                S  S  S  V  T  F  F        S  S  S  S
                E  E  E  B  S  E  E        E  E  E  E  T
                R  R  R  L  T  I  R  D     R  R  R  R  L  P  P  D
                V  V  V  A  A  L  R  X  G  V  V  V  V  O  N  N  O
                E  E  E  N  R  D  O  D  N  E  E  E  E  A  I  I  N
                D  D  D  K  T  L  R  Y  D  D  D  D  D  D  2  1  E
              ----------------------------------------------------_
             /  9  8  7  6  5  4  3  2  1 68 67 66 65 64 63 62 61   |
      CXOUT | 10                                               60 | RESERVED
      CYOUT | 11                                               59 | COUT3
     ACQOUT | 12                                               58 | COUT2
   RESERVED | 13                                               57 | COUT1
       DCLK | 14                                               56 | VBLANKL
        DXL | 15                                               55 | VALPT
       DYLi | 16                                               54 | TSPHI
      ERROR | 17                                               53 | start
        Vcc | 18                                               52 | Vcc
      FDONE | 19                                               51 | OUTLAT
     FEPHI2 | 20                                               50 | OLATCH
      FIELD | 21                                               49 | INVL
    HBLANKL | 22                                               48 | HSYNC1
        Gnd | 23                                               47 | BIT5
        Gnd | 24                                               46 | ACQIN
        Gnd | 25                                               45 | Gnd
        Gnd | 26                                               44 | Gnd
            |_ 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43  _|
              ----------------------------------------------------
                G  G  F  H  R  R  R  R  G  G  G  G  H  H  I  L  S
                n  n  R  S  A  E  E  E  N  n  n  n  B  B  N  I  T
                d  d  E  Y  M  S  S  S  D  d  d  d  L  L  V  N  A
                      S  N  W  E  E  E              A  A     E  R
                      E  C  E  R  R  R              N  N     S  T
                      T        V  V  V              K  K     T
                               E  E  E                 B
                               D  D  D                 L
```