DESIGN OF A VLSI CONVOLUTION SYSTEM FOR IMAGE PROCESSING

Damian Daniel Haule

B.Sc. (Eng.), (University of Dar es Salaam, Tanzania)

Department of Electrical Engineering

McGill University

A thesis submitted to the Faculty of Graduate Studies and Research

in partial fulfillment of the requirements for the degree of

Master of Engineering (M.Eng.)

March, 1990

Abstract

One of the most commonly used algorithms in image processing is certainly the two dimensional (2-D) convolution. It is also widely used in graphics, filtering applications, linear, shift-invariant filters, like in edge detection, noise reduction, etc. The main problem with 2-D operations is that they require intensive computations even for small image windows. This large computational requirements takes minutes or hours if implemented in software and it grows very rapidly with the convolution kernel size.

One solution to this problem is to build a dedicated hardware system that can perform the convolution calculation in a highly regular and parallel fashion. This hardware approach is followed and a specialized VLSI convolution chip is designed and fabricated to perform a part of the entire convolution operation. The chip is to be used in an autonomous robot vision system, hence novel features are included for the detection and minimization of convolution errors especially overflows.

This thesis describes the design of a VLSI convolution system for image processing in a robot vision system. A dedicated hardware system which performs a 2-D convolution operation on a 512x512 pixel image in less than a second for a 9x9 window size is presented. The entire system fits on a single **MULTIBUS** card. The design requirements included the specialized VLSI convolver chip as the basic processing element. Furthermore, the system is **DMA** driven so that CPU independent processing is possible. Finally, the card includes a MULTIBUS compatible interface allowing it to be supported in the **INTEL iRMX-286** operating system environment.

Résumé

L'une des techniques les plus connues dans traitement de l'image est évidemment la convolution en deux dimensions (2-D). Elle est aussi utilisée fréquemment dans graphique par ordinateur, techniques de filtrage, filtres linéaires et "invariant à la translation", dans le but de détecter des contours, réduire les bruits, etc... Un problème majeur de ces opérations en 2-D est la consommation énorme du temps de calcul, même pour des petites fenêtres. Le temps de calcul dans les filtres croît très rapidement avec la grandeur de noyau de convolution.

Une solution pour ce problème est de concevoir un système qui peut accomplir des calculs de convolution d'une manière précise et parallèle. L'approche optimale est de concevoir un circuit-intégré spécialisé : il s'agit de fabriquer un processeur pour la convolution, celui-ci peut accomplir une partie de l'opération entière. Ce processeur est utilisé dans un système de vision robotique, dans lequel les nouvelles caractéristiques sont nécessaire pour détecter des erreurs de convolution et en minimiser, particulièrement les "dépassement de capacité".

Cette thèse décrit la conception d'un système VLSI spécialisant dans les calculs de convolution, pour traitement de l'image dans un système de vision robotique. Ce système "hardware" peut accomplir la convolution en 2-D sur une image 512x512 éléments en moins d'une seconde pour une fenêtre 9x9. Le système en entier est installé sur une carte qui peut communiquer avec le **MULTIBUS**. La conception exige le processeur VLSI de convolution comme un élément de traitement de base. De plus, le système doit être conduit par **DMA** pour que le CPU soit indépendent. Finalement, la carte doit communiquer avec MULTIBUS pour pouvoir fonctionner sur le système d'exploitaion Intel **iRMX-286**.

Acknowledgements

Many people, in one way or another, have assisted me in writing and preparing this thesis. At the outset I would like to thank my advisor, Dr. A.S. Malowany, for introducing me to the subject of Computer vision and for his guidance and encouragement while doing this research. Lastly, but not least, he was instrumental in influencing the organization and results of this thesis.

Thanks are also due to Dr. N. Rumin and Dr. J. Rajski, who made the access to the VLSI Laboratory's design facilities possible as well as introducing the material provided in the VLSI Design I course and VLSI Testing course respectively.

Several colleague students have helped me in writing programs, preparing data laboratory experiments, picture output, carrying out relevant library research, etc. In this regard , I would like to thank J.F. Côté, C. Collet, L. Martin and D. Melanson. It is my pleasure to give particular credit to all those who, in one way or another, have given their encouragement and dedicated effort toward this thesis realization.

Finally, the author would like to thank the Department of Electrical Engineering, University of Dar es Salaam, Tanzania for granting him an educational leave of absence and the Association of Universities and Colleges of Canada (AUCC) for their financial support during his stay at McGill University, Montreal, Canada for the Master of Engineering (M.Eng.) degree programme of study. The financial supports of NSERC and CMC are also gratefully acknowledged.

Damian D. Haule

Table of contents

Table of Figures

List of Tables

1. INTRODUCTION

In this chapter, problems and requirements associated with computer vision as applied to robot systems are introduced. The computer hardware organization of the Computer Vision and Robotics Laboratory (CVaRL) of McGill Research Center for Intelligent Machines (McRCIM) is given. Finally, a brief overview of this thesis research is presented.

1.1 Computer Vision and Robotics

The use of robots in industries has become increasingly important to improve productivity and reduce production costs. Robots are used to perform repetitive tasks such as pick and place, spot welding, spray painting, etc. The precise definition of a robot was given by K.S. Fu[32] as follows:-

"A robot is a **reprogrammable**, multifunctional **manipulator** designed to move material, parts, tools, or specialized devices through variable programmed motions for the **performance of a variety of tasks**".

The above highlighted keywords could be summarized with the word **flexibility**. Most robots require the precise positioning of the object to be processed due to lack of sensory feedback from the work environment (except for the robot's joint encoders). Thus, the flexibility of robotic workstations is reduced by this characteristic.

At present, most researchers are working to introduce sensory feedback for robot control and for parts inspection such as tactile sensors[43],[78],[98],[26], proximity detectors[21],[84], etc. Computer vision is among the most flexible sensory feedback being investigated. It permits performing motion planning and control[70],[42],[32],[33], adjusting the planned motions to obtain the exact position and orientation of parts being moved, as well as parts inspection[23],[38],[80],[69].

Commercial systems, with limited vision capabilities, which can recognize images of one or more non-overlapping objects with the position and orientation in less than one second include the VS-100 and VS-110 from Machine Intelligence Corporation and the Octek Vision Module. Movich[68] describes the application of such a system for drilling and riveting of aircraft parts, Reinhold[80] applied the system for the automatic inspection of sheet metal parts, while Dillman[29] was involved in the quality control of label applications.

Binary imaging systems do not fare well in many applications such as inspection, unpacking, etc. This is because contrast is not sufficiently high or cannot be controlled, or the applications required more than the outline. In these cases, it becomes necessary to use grey level processing, in which pixels may take any value within a range, each representing a different intensity in the original image. The advantage of grey-scale imagery is the amount of detail the image contains about the world. The object itself, and not only the outline as is the case for binary images, may be studied and inspected. Neverthless, grey-scale images are more costly to acquire, transfer, store, access and process to extract discriminating features and take longer to process with even more complicated algorithms.

Many approaches have been proposed to reduce the computation time to an acceptable level[22],[60]. T.J. Fountain et al[30],[31] describe CLIP7A and CLIP IV, a cellular logic processor with 96x96 computational elements. K.E. Batcher[6] developed a massively parallel processor (MPP) with 128x128 bit-serial computational elements, while H.J. Siegel et al[87] describe a system in which many microprocessors communicate through a local interconnection network. Investigation of three-dimensional structures is also taking place, for example pyramidal computers[27],[77],[72],[18],[90],[2]. More specialized systems for robotic applications are also designed such as with systolic arrays[58],[28],[17],[49] and with the concept of wavefront computing[53],[45],[64]. Processors developed especially for robot vision systems must include multi-microprocessors operating in parallel to reduce the computation time[57],[58],[84]. Such

systems are needed before grey level image processing techniques may be used. Future attention of researchers will shift toward multi-spectral imagery, such as color images[63],[15], and 3-dimensional signal processing that will use stereoscopic[95],[36],[27] or structured light techniques[41],[47] or depth maps[75],[74],[86] to increase the robot system's world perception to include depth.

Research studies in the department of Electrical Engineering at McGill University in the Computer Vision and Robotics Laboratory (CVaRL) include the use of image processing and computer vision techniques for robot assembly and repair. This study continues to address the implementation of a flexible workstation for the automatic assembly and repair of hybrid and printed circuits[35],[60],[66],[65].

Figures 1.1 and 1.2 show the general organization of the laboratory's research computer facilities. They are based on VAX 32-bit mini-computers, SUN and Micro-VAX computers, all running UNIX 4.2, VAX/VMS and/or Eunice, a Unix emulator. All computers are connected through a local area network based on Ethernet and the TCP/IP protocol.

Main areas of research include computer vision, visual inspection of traces, solder joints, collision avoidance, motion planning, control algorithms for robots, etc. Most of these tasks involve a major use of visual feedback. More details are given in the 1988 CVaRL Progress Report[76].

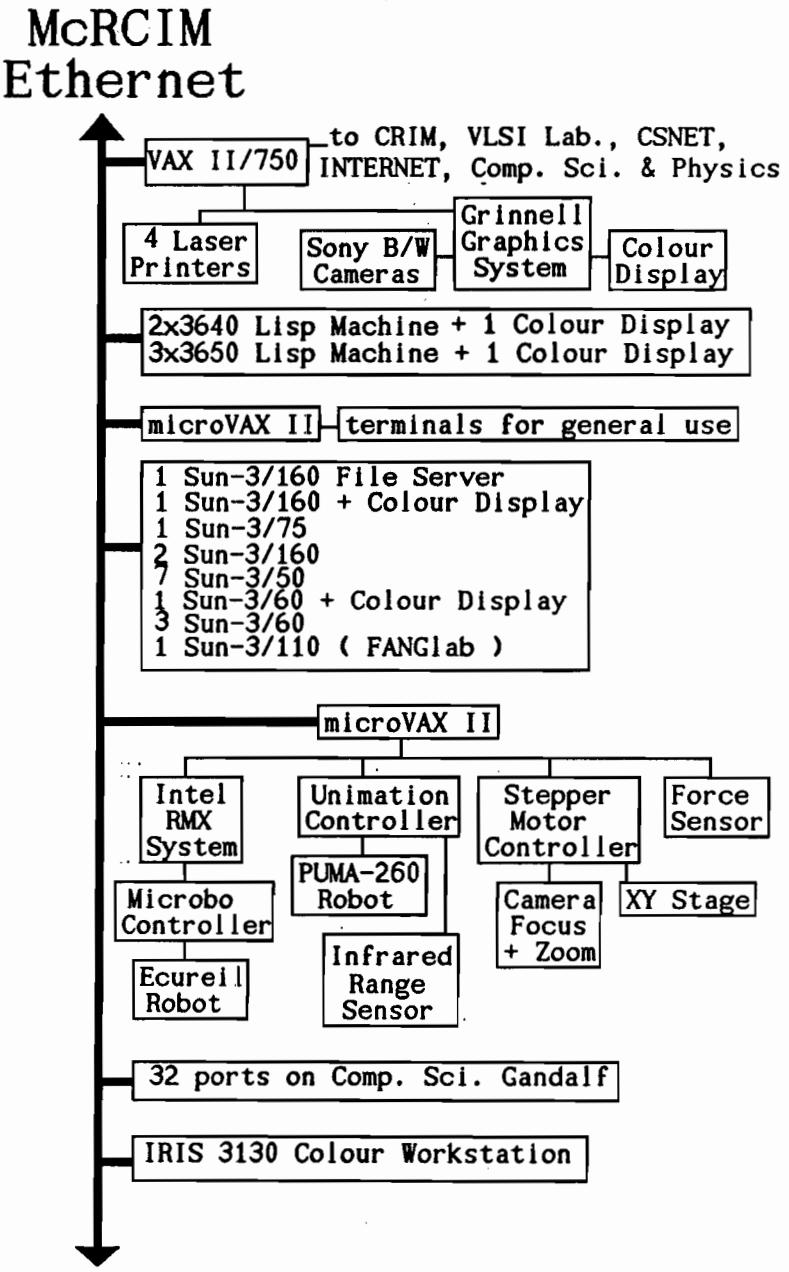


Figure 1.1 Organization of CVaRL Research Facilities (Fig. 1 of 2)

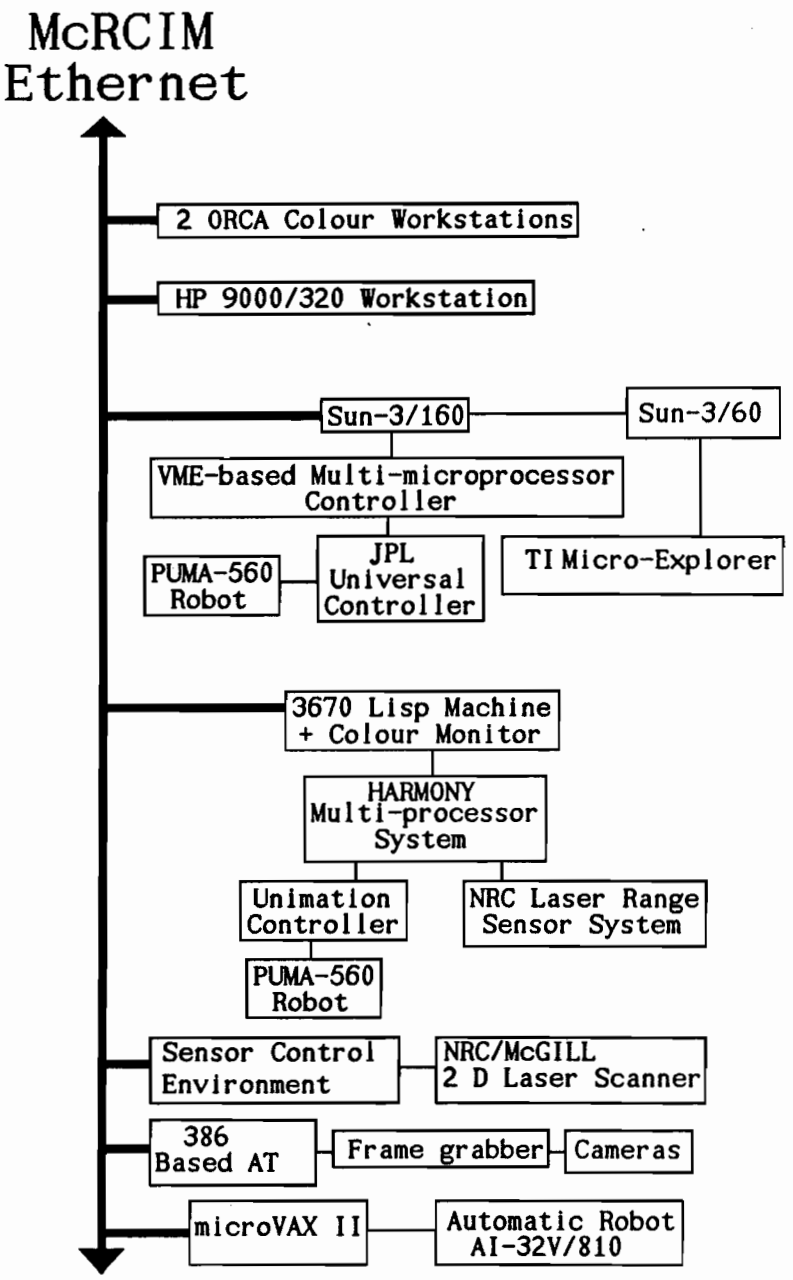


Figure 1.2 Organization of CVaRL Research Facilities (Fig. 2 of 2)

1.2 Thesis Overview

This thesis describes the implementation of a high-speed 2-dimensional hardware convolution architecture based on VLSI systolic arrays for image processing applications[44]. It addresses the issues of designing a special purpose systolic image convolution processor meant for use in a robot vision system. An architecture for the parallel processing of the generalized 2-D convolution is also presented.

Firstly, a VLSI convolution chip was designed to accommodate various convolution window sizes. The number of coefficients being handled is directly proportional to the number of systolic computing elements (processors) used. In this design three such processors are configured on one VLSI chip. Signed coefficients and unsigned data of 8-bits are supported. All processing and inter-processor communications are performed bit-serially. The chip design incorporates error detection during convolution and overflow avoidance techniques are proposed for maximum system autonomy. The chip is estimated to operate at a maximum frequency of 20 MHz. under normal conditions as discussed in Chapter 8 when the fabricated VLSI convolution processor was tested.

The designed VLSI convolution processor cell consists of a shift-register pipeline, an 8 bit array of multipliers and a tree of adders. The image data enter the processor in a raster scan format and are stored and shifted in the pipeline. The multiplier array takes data from the pipeline serially, does the multiplication with the stored coefficient, and then sends the partial products to the adder tree to complete the computation of the partial sum.

The increasing resolution in modern image technology allows for improved precision in computer vision but, increases the time-complexity of image processing. Image processing frequently uses two dimensional convolution[59],[51],[88], a process requiring over twenty mil-

lion additions and multiplications for a single 9x9 window size convolution on a 512x512 picture frame . For robotic purposes, the convolution results are needed within a reasonably short amount of time so as to allow the robot to react in real-time. A 9x9 convolution requires close to an hour of computation time on a Sun workstation in our computer laboratory, a delay that is much too high for robotic vision purposes. This thesis also describes the design of a convolution processor card that performs a 9x9 window size convolution on a 512x512 picture frame in less than a second.

Most image processing algorithms are very computation intensive[8]. In fact, many real-time image processing throughput rates outstrip current parallel architectures. For example, low level operations, such as filtering or enhancement, typically require the order of some tens of millions of machine instructions per image. Image processing uses a wide variety of mathematical techniques, dominated by convolution/correlation filtering, transform techniques, etc. all of which require intensive computation. Thus image processing is a major driving force in the development of faster and more powerful parallel architectures.

The key to successful design of a VLSI convolution processor lies in a cohesive exploitation of the applicational, algorithmic, and architectural aspects[13],[39]. Hence, this thesis demonstrates the regular and cohesive design of a highly parallel image convolver hardware system, using a designed VLSI convolution chip or processor. The systolic convolution cell is designed to multiply a pixel intensity by a given coefficient and to add this product to a corresponding partial sum. Twenty seven systolic cells are arranged in a pipeline architecture to realize a high performance convolution board or card for the convolution window size of 3x9.

The new Intel 82258 Advanced DMA Controller[47] is used to perform each pixel transfer to and from the host computer's memory. Due to the DMA's software programmability,

pictures of any size can be processed as discussed in more detail in Chapter 6. The twenty seven systolic cells incorporated into the circuit constitute a basic convolution window size of 3x9. The image data for these three rows is supplied by three concurrent DMA transactions while the convolved output is handled by the fourth channel on the DMA chip. This cycle is repeated three times with different coefficients to achieve a 9x9 window size. A fourth pass is used to combine the partial convolutions into a final image. Since each pass requires only 200 milliseconds to process the entire image, the 9x9 convolution is obtained in less than one second[24]. Chapter 6 briefly describes the DMA subsystem operation.

The convolution board was assembled on a 36 column MULTIBUS bus card and is being installed on an Intel System 310 running iRMX-286 real-time multitasking operating system for testing. This implementation will offer fast and reliable computer vision for robotic applications. Moreover, its unlimited flexibility in terms of frame and window size will, in the future, allow it to support even higher resolution image technology or the flexibility of electronic windowing on the new selective scan camera systems such as the CIDTEC model 2250 solid state camera[3],[5],[12]

2. CONVOLUTION COMPUTATIONS

Continuous-time convolution is mainly of theoretical importance, while discrete-time convolution has many practical applications. In particular, it may be used to implement linear, shift-invariant filters for noise-reduction, edge enhancement and many more image-processing applications.

2.1 Linear Convolution Operator

The 2-D linear convolution of two signals $x_1(m,n)$ and $x_2(m,n)$ is defined as:

$$\begin{aligned} \mathbf{y(m,n)} &= x_1(m,n) * x_2(m,n) \\ &= \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} x_1(i,j)\, x_2(m-i,n-j) \\ &= \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} x_2(i,j)\, x_1(m-i,n-j) \\ &= x_2(m,n) * x_1(m,n) \end{aligned} \tag{2.1}$$

Where "* " denotes the convolution operator and convolution order is unimportant.

Representing the input signal $x(m,n)$ by a sequence of impulses we get:

$$\mathbf{x(m,n)} = \sum_{\mathbf{i}=-\infty}^{+\infty} \sum_{\mathbf{j}=-\infty}^{+\infty} \mathbf{x(i,j)}\delta\mathbf{(m-i,n-j)} \tag{2.2}$$

Where

$$\delta\mathbf{(m,n)} = \begin{cases} 1 & \textbf{if m=n=0} \\ 0 & \textbf{otherwise} \end{cases} \tag{2.3}$$

For linear, shift-invariant system with output $y(m,n)$, the system's output is given by:

$$\begin{aligned} \mathbf{y(m,n)} &= T[x(m,n)] \\ &= T[\sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} x(i,j)\delta(m-i,n-j)] \end{aligned} \tag{2.4}$$

By the linearity property we get[81]:

$$\mathbf{y(m,n)} = \sum_{\mathbf{i}=-\infty}^{+\infty} \sum_{\mathbf{j}=-\infty}^{+\infty} \mathbf{x(i,j)}\ \mathbf{T}[\delta(\mathbf{m}-\mathbf{i},\mathbf{n}-\mathbf{j})] \tag{2.5}$$

By the shift-invariance property we get[81]:

$$\mathbf{y(m,n)} = \sum_{\mathbf{i}=-\infty}^{+\infty} \sum_{\mathbf{j}=-\infty}^{+\infty} \mathbf{x(i,j)}\ \mathbf{h(m-i,n-j)} \tag{2.6}$$

where $\mathbf{h(m,n)}$ is the system's impulse response.

Hence we have:

$$\mathbf{y(m,n)} = \mathbf{x(m,n)} * \mathbf{h(m,n)} \tag{2.7}$$

i.e. convolution between $x(m,n)$ and $h(m,n)$.

The response of any linear, shift-invariant system may be expressed as the convolution of the input signal $x(m,n)$ along with the system's impulse response $h(m,n)$ called the **convolution filter or kernel.**

The input signal is an image of limited dimensions. This has the advantage of limiting the summation range to that of the dimensions of the image and, hence, the convolution will always be absolutely summable as long as the coefficients of the impulse response and the pixel intensities are finite.

2.2 Circular Convolution Operator

Circular convolution[81] is another approach to be used. Assuming two signals $x_1(m,n)$ and $x_2(m,n)$, each of size MxN, and their discrete fourier transform (DFT) be $X_1(u,v)$ and $X_2(u,v)$ respectively, then, the circular convolution is defined in such a way that:

$$\mathbf{x_3(m,n)} = \mathbf{x_1(m,n)} \bigotimes \mathbf{x_2(m,n)} \tag{2.8}$$

which implies:

$$\mathbf{X_3(u,v)} \;=\; \mathbf{X_1(u,v)\;X_2(u,v)} \tag{2.9}$$

where $\bigotimes$ denotes the circular convolution operator. Convolution is simply multiplication of respective DFT's in frequency domain.

It can be shown that[8]:

$$\mathbf{x_3(m,n)} \;=\; \left\{ \sum_{\mathbf{k=0}}^{\mathbf{M-1}} \sum_{\mathbf{l=0}}^{\mathbf{N-1}} \mathbf{x_1([k]_M,[l]_N)\;x_2([k-m]_M,[l-n]_N)} \right\} \mathbf{R_{M,N}(m,n)} \tag{2.10}$$

where $[x]_M$ is x modulo M and

$$\mathbf{R_{M,N}} \;=\; \begin{cases} 1 & \textbf{if } 0 \leq m \leq M-1 \textit{ and } 0 \leq n \leq N-1 \\ 0 & \textbf{otherwise} \end{cases} \tag{2.11}$$

The circular convolution may be interpreted as creating periodic signals by replicating $x_1(m,n)$ and $x_2(m,n)$ in both directions, and computing the linear convolution over one period. The result is then obtained by windowing the resulting signal to the original MxN dimensions. The circular convolution is also absolutely summable only when both the amplitude of both input signals are finite because of the limited summation range. It can also be shown that:

$$\mathbf{y(m,n)} \;=\; \mathbf{x_1(m,n)} \bigotimes \mathbf{x_2(m,n)} \;=\; \mathbf{x_2(m,n)} \bigotimes \mathbf{x_1(m,n)} \tag{2.12}$$

i.e. again convolution order doesn't matter.

2.3 1-D Convolution

A convolution operation simply involves multiplying and adding. The actual convolution operation can be implemented in hardware by using a specially designed VLSI convolution chip. The 1-D convolution chip designed can be used as a basic cell for higher dimensional processing. Thus, a 2-D convolution can be performed by cascading basic cells as building blocks. Each basic cell is self sufficient and its output is equal to the convolved input. The basic 1-D convolution chip is as shown in Figure 2.1[51].

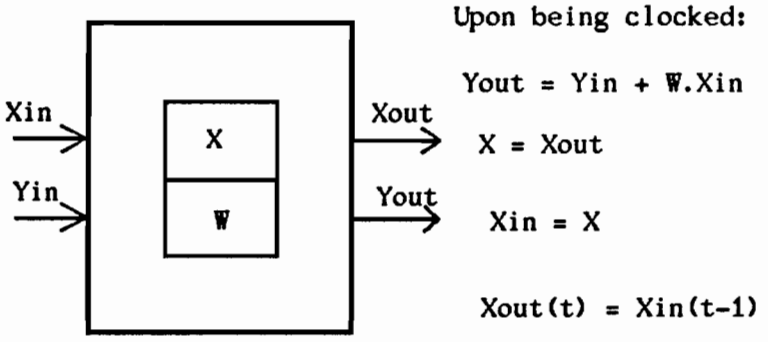


Figure 2.1 Basic 1-D Convolution Cell

where:

X_{in} = input data

Y_{in} = partial result from previous cell

X_{out} = delayed out data

Y_{out} = convolved output (partial result for the next cell).

Basic characteristics[51],[97] of the 1-D convolution chip can be summarized as:

(a) it has serial inputs and serial outputs,

(b) each cell is pre-loaded with weights,

(c) input data is delayed within the cell to allow pipelining,

(d) cells are cascaded by connecting inputs to the outputs of the previous cell,

(e) a load input enables the loading of weights and

(f) a chip enable input enables the convolution computation.

2.4 2-D Convolution

One of the most compute-intensive tasks in image processing is the 2-D convolution problem[8],[18]. Consider the problem of convolving an mxm kernel with an nxn image, where $m \leq n$. Assume that m is not large enough to make an FFT-based solution cost-effective. Then the straightforward method of solving the problem will require $O(m^2n^2)$ operations.

Several systolic devices for the 2-D convolution problem have been proposed[51],[97],[57]. Suppose that such a device can only handle kernels of size kxk, where $k \leq m$. Then the mxm kernel has to be decomposed into subkernels of size kxk. Consider the algorithm that slides the kernel from left to right until the right boundary of the image is reached. Then slide down one row of image and do the same, until the bottom of the image is reached.

It is easy to see that kernel elements have to enter the device $\mathbf{O(n)}$ times and entries of image $\mathbf{O(m^2/k)}$ times. Thus, the I/O requirement for the kernel is the highest, that for the input image is the second, and that for the resulting image is the lowest. A standard 2-D

convolution equation is given by[73]:

$$y_{r,s} = \sum_{i=1}^{k}\sum_{j=1}^{k} W_{i,j}.X_{r+i-k+1,\ s+j-k+1} \tag{2.13}$$

where:

w = weights pre-loaded in each cell

X = pixel information in grey scale

k = size of convolution kernel

$y_{r,s}$ = convolution result for pixel r, s.

A major drawback in 2-D convolution is that of reading each row k times in order to convolve the whole image. A standard way to correct this problem is to use a **Systolic Convolver** and to delay the row data as required before feeding it to the adjacent rows of the systolic design. However, this row delay implementation becomes cumbersome and costly in hardware implementation when processing images of variable dimensions. The next chapter outlines alternative systolic designs for convolution-like processing.

3. SYSTOLIC CONVOLUTION ARCHITECTURES

3.1 Definition and Principle of a Systolic Architecture

A Systolic architecture[50] is a design methodology in which many functional modules can be mapped into silicon cost-effectively, i.e. a powerful method for implementing cost-effective silicon subroutines for computations such as pattern matching, error correcting, data base processing, signal and image processing, etc. It results in cost-effective, high-performance special purpose systems applicable to a wide range of problems[14],[96] .

In a systolic system data flows from the computer memory in a rhythmic fashion, passing through many processing elements (PE) before it returns to memory (Cf. blood circulation to and from the heart) as shown in Figure 3.1. Data flow may be at multiple speeds in multiple directions, both inputs and (partial) results flow, whereas only results flow in classical pipelined systems. Each data is used multiple times, hence it can speed up execution of compute-bound problems without increasing I/O requirements. It is easy to implement because of its regularity and to reconfigure (to meet various outside constraints) because of its modularity.

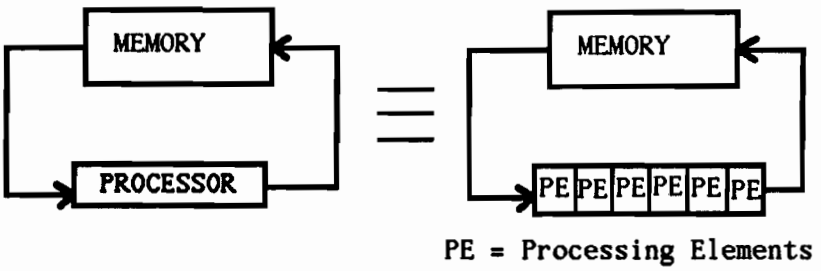


Figure 3.1 Basic Principle of Systolic Architecture

3.2 Key Architectural Issues

In designing a special-purpose system, three phases[11],[39] can be identified:

1. Task definition phase,

2. Design phase and

3. Implementation phase.

In task definition phase, some system performance bottleneck is identified, and a decision on whether or not to resolve it with special-purpose hardware is made. In the design phase, one has to consider three main items:

(i). The design process. Simplicity and regularity of a design influence its cost-effectiveness. There is a trade-off between nonrecurring (design) and recurring (parts) costs. A basic strategy is to decompose the design into a few types of simple substructures or building blocks, which are used repetitively with simple interfaces. The design should be modular and adjustable to meet various performance goals.

(ii). Concurrency and Communication. Two possible ways to build a fast computer system are either to use fast components or to use concurrency, i.e. use of many processing elements. The degree of concurrency is largely determined by the underlying algorithm. Massive parallelism can be achieved if the algorithm is designed to introduce high degrees of **pipelining and multiprocessing.** When a large number of processing elements work simultaneously, coordination and communication become significant. Hence, design algorithms[8] that support high degrees of concurrency, while employing only simple, regular communication and control to enable efficient implementation are

necessary.

(iii). Balancing Computation with I/O. A computation rate should balance the available I/O bandwidth with the host computer, memory, real-time device, etc.

Systolic architectures consist of a set of interconnected **cells**[52],[57], each capable of performing some simple operation. Cells are interconnected to form a systolic array or a systolic tree. Information flows between cells in a pipelined fashion, and communication with the outside world occurs only at the **boundary cells** . Compute-bound architectures (when computing operations take longer than I/O transactions) and I/O-bound architectures (when the I/O transactions take longer than the computing operations) require different implementation strategies.

Systolic designs in one- or two-dimensional arrays are commonly used for the regular, compute-bound architectures. In signal and image processing applications[17], systolic designs have been proposed for FIR and IIR filtering[81], and 1-D convolution, 2-D convolution and correlation[100], Discrete Fourier Transform (DFT), Interpolation, 1-D and 2-D median filtering[83] and Geometric warping[19],[89].

In Matrix Arithmetic applications[34], existing systolic designs include matrix-vector multiplication, matrix-matrix multiplication, matrix triangularization (solution of linear systems, matrix conversion), QR-decomposition (eigenvalue, least squares computations), solution of triangular, linear systems, etc. In non-numeric applications there are designs for data structures (stack and queue), searching, priority queue, and sorting, graph algorithms (transitive closure, minimum spanning trees, geometric algorithms), convexity hull generation, language recognition (string matching and regular expression), dynamic programming, polynomial algorithms (multiplication and division, greatest common divisors), relational

data-base operations, etc.

3.3 Systolic Arrays for Convolution-like Computations

The convolution problem is defined as follows[51]:

Given the sequence of weights $(w_1, w_2, ..., w_k)$ and the input sequence $(x_1, x_2, ..., x_n)$, **compute** the result sequence $(y_1, y_2, ..., y_{n+1-k})$ defined by:

$$y_i = w_1 x_i + w_2 x_{i+1} + ... + w_k x_{i+k-1} \tag{3.1}$$

The convolution problem is also called the FIR - filtering problem or the correlation problem, i.e. combining two data-streams, w_i's and $x_i's$ in a certain manner to form a resultant data-stream of y_i's. The convolution problem is compute-bound, since each input x_i is to be multiplied by each of the k weights.

3.3.1 Semi-Systolic Convolution Arrays

Semi-systolic convolution arrays have global data communication. If an x_i, once brought out from the memory, is broadcast to a number of cells, then the same x_i can be used by all the cells, i.e. multiple use of each input element. The opposite of broadcasting is **fan-in**, through which data items from a number of cells can be collected. Different systolic designs that utilize broadcasting and fan-in are as follows:

1. **Broadcast inputs, move results, weights stay:**

   Figure 3.2 depicts this type of design. Weights are pre-loaded to the cells, one at each cell, and stay at the cells throughout the computation while partial results y_i move systolically from cell to cell in the left-to-right direction.

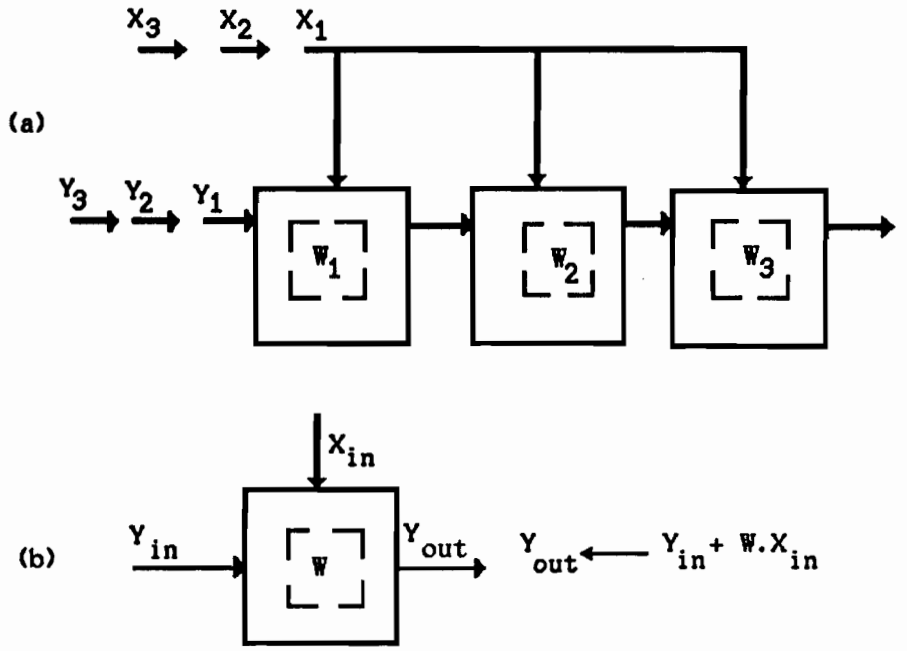


Figure 3.2 (a) Systolic convolution array and (b) Cell, where w_i's stay, x_i's are broadcast, and y_i's move systolically.

2. Broadcast inputs, move weights, results stay:

This design is as shown in Figure 3.3. Each y_i stays at a cell to accumulate its terms. The weights circulate around the array of cells. Its disadvantage is that a separate bus is required for collecting outputs and its advantage is that the systolic path for moving w_i's is less wider than that of y_i's since for numerical accuracy y_i's carry more bits than w_i's.

3. Fan-in results, move inputs, weights stay:

Figure 3.4 shows this type of design. Consider the vector of weights $(w_k, w_{k-1}, ..., w_1)$ as fixed in space and input vector $(x_n, x_{n-1}, ..., x_1)$ as sliding over the weights in the left-to-right direction. Weights are pre-loaded to the cells and stay throughout the

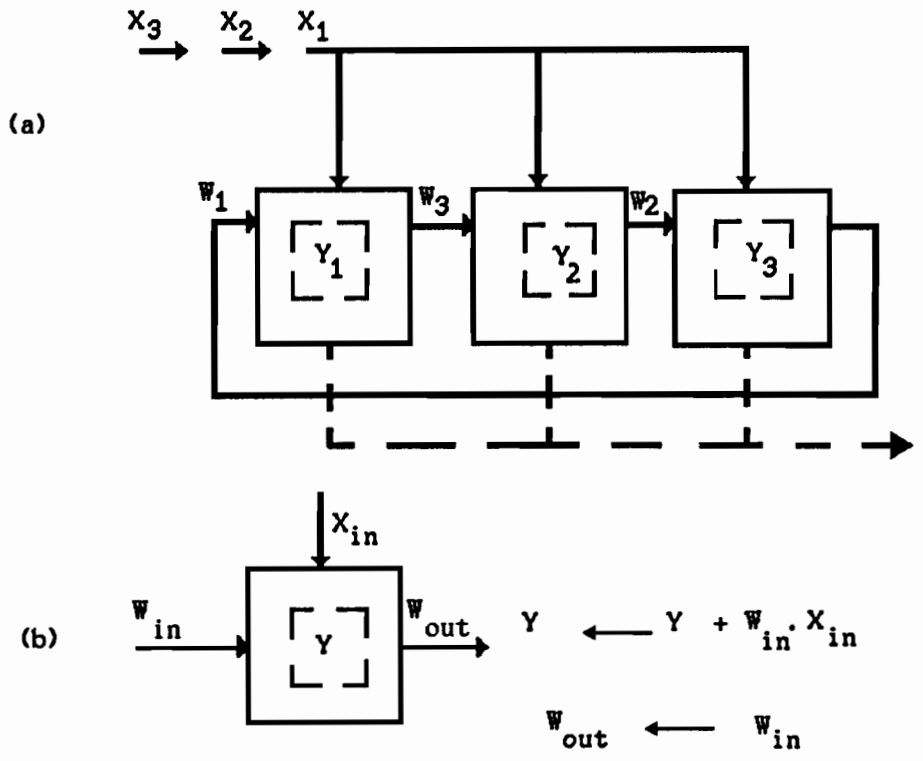


Figure 3.3 (a) Systolic convolution array and (b) Cell, where y_i's stay, w_i's move systolically, and x_i's are broadcast.

computation while results of multiplications are fanned-in and summed using an adder to form y_i.

3.3.2 Pure-Systolic Convolution Arrays

Pure-systolic convolution arrays have no global data communication. Global broadcasting or fan-in solves the I/O bottleneck problem, but implementing it in a modular, expandable way presents another problem, i.e. providing (or collecting) a data item to (or from) all the cells of a systolic array, during each cycle, requires the use of a bus or some sort of tree like network. Hence pure-systolic convolution arrays without global data communication do exist which can be extended to include an arbitrarily large number of cells. Different types

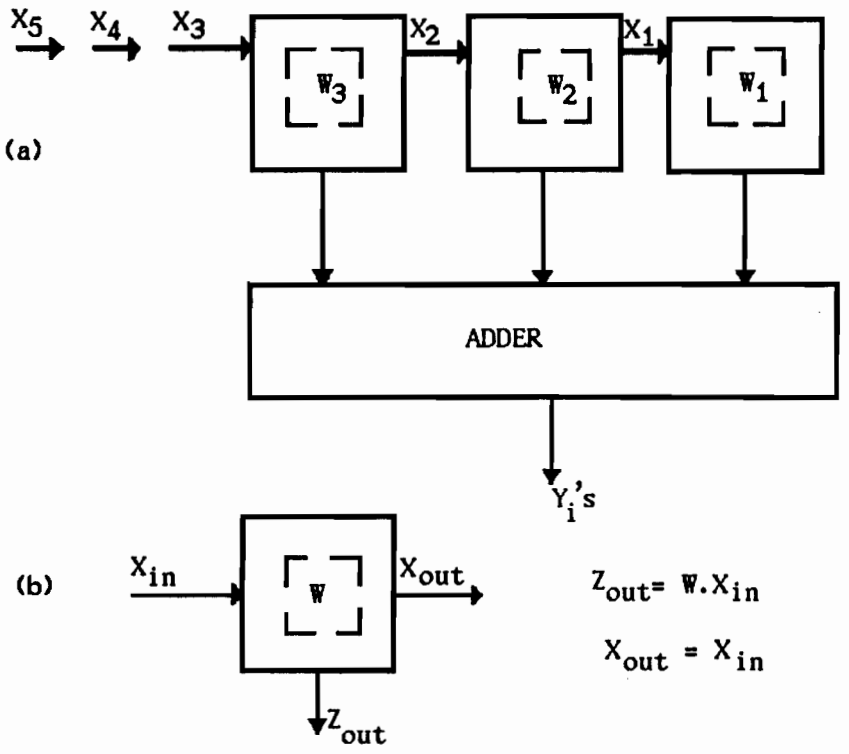


Figure 3.4 (a) Systolic convolution array and (b) Cell, where w_i's stay, x_i's move systolically, and y_i's are formed through the fan-in of results from all the cells.

of designs exist for pure-systolic convolution arrays:

1. **Results stay, inputs and weights move in opposite directions:**

Figure 3.5 shows this design. Each partial result y_i stays at a cell to accumulate its terms. The x_i's and w_i's move systolically in opposite directions such that when an x meets a w at a cell, they are multiplied. A systolic output path is sufficient for collecting output from cells. The advantage of this design is that it does not require a bus or global network. To ensure that each x_i is able to meet every w_i, consecutive x_i's are separated by two cycle times and so are the w_i's.

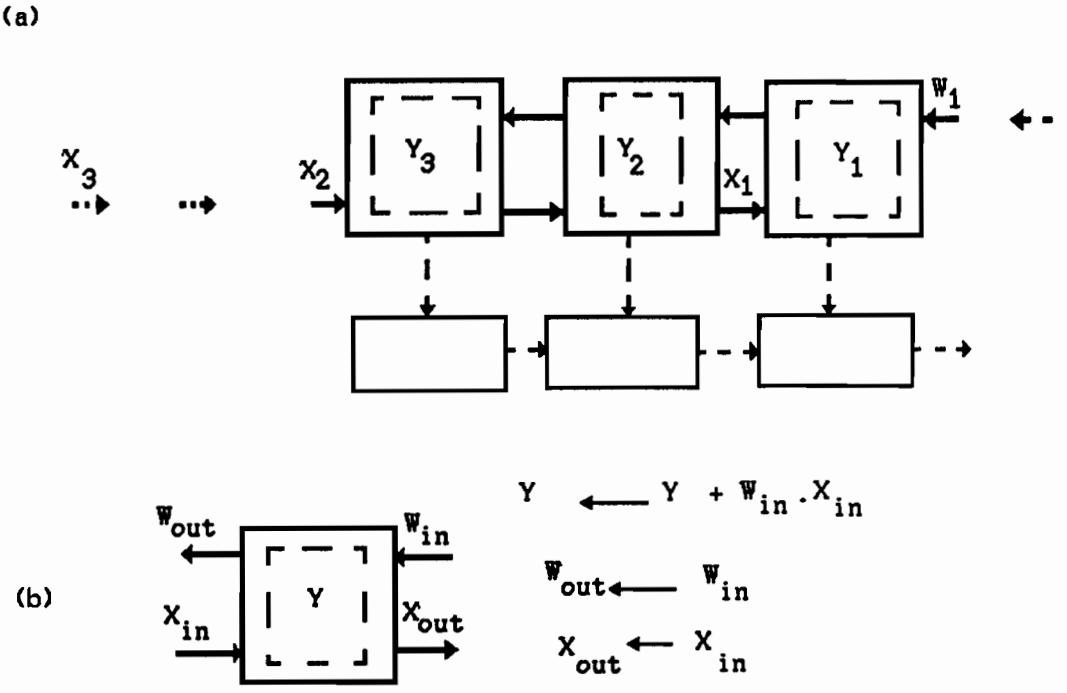


Figure 3.5 (a) Systolic convolution array and (b) Cell, where y_i's stay, x_i's and w_i's move in opposite directions systolically.

2. Results stay, inputs and weights move in the same direction but at different speeds:

This type of design is shown in Figure 3.6. Both the x and w data streams move from left to right systolically but the x_i's move twice as fast as the w_i's, i.e. each w_i stays inside every cell for one extra cycle. In this case, the multiplier-accumulator hardware is used effectively. The first weight w_1 is associated with a tag bit that signals the accumulator to output and resets its contents. All the cells work all the time when performing a single convolution but it requires an additional register in each cell to hold a w value. It is also possible to have a dual design whereby w_i's move twice as fast as x_i's.

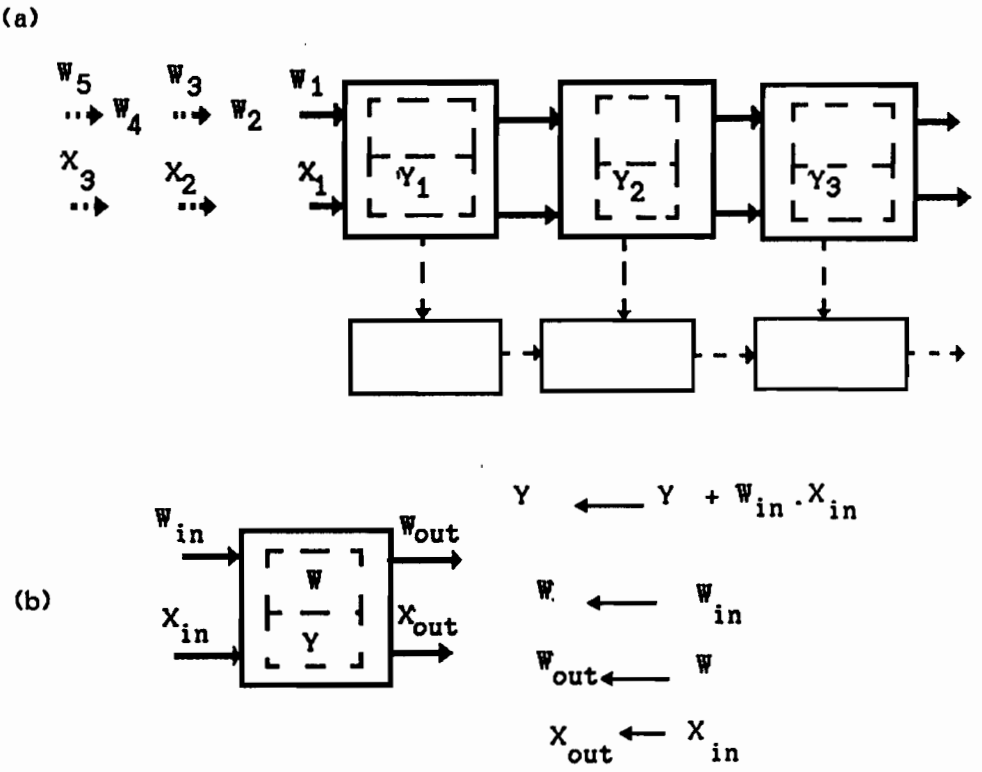


Figure 3.6 (a) Systolic convolution array and (b) Cell, where y_i's stay, x_i's and w_i's both move systolically in the same direction but at different speeds.

3. Weights stay, inputs and results move in opposite directions:

Figure 3.7 shows this design. Weights stay, one at each cell, but results and inputs move systolically. Although they are not geared to the most effective use of available multiplier-accumulator hardware, they are potentially more efficient than other designs so far discussed. Because the same set of weights is used for computing all the y_i's and different sets of the x_i's are used , it is natural to have the w_i's preloaded to the cells and stay there, and let the x_i's and the y_i's move along the array. It is proposed as a special case of systolic filtering array. The main advantages of this design is that it can be naturally extended to perform recursive filtering. Its drawback is that only one half of the cells work at any time.

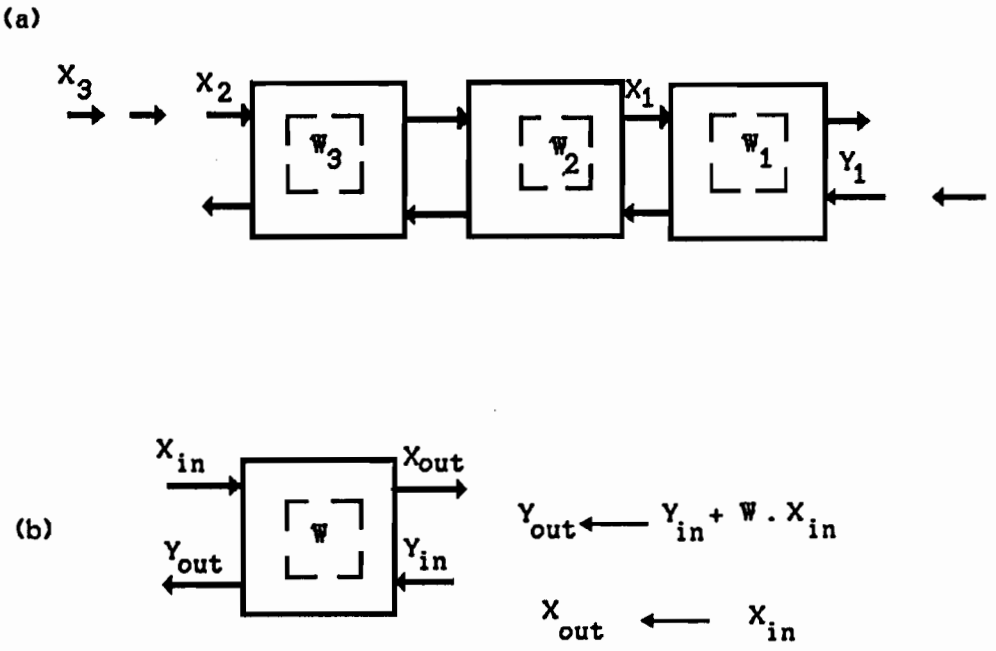


Figure 3.7 (a) Systolic convolution array and (b) Cell, where w_i's stay, x_i's and y_i's move systolically in opposite directions.

4. Weights stay, inputs and results move in the same direction but at different speeds:

This type of design is shown in Figure 3.8. Weights stay, one at each cell, but results and inputs move systolically at different speeds. Cells work all the time but the Y outputs travel twice as fast as the X stream. The row output takes place k cycles after ts input starts entering the left-most cell of the systolic array (where k is the number of systolic cells being chained). This design can be extended to implement 2-D convolutions[50],[51]. The design has a dual version for which the x_i's move twice as fast as the y_i's. The design of Figure 3.8 was selected for the VLSI systolic convolution processor mainly because of its simple control strategy requirements as will be described in detail in the following chapters.

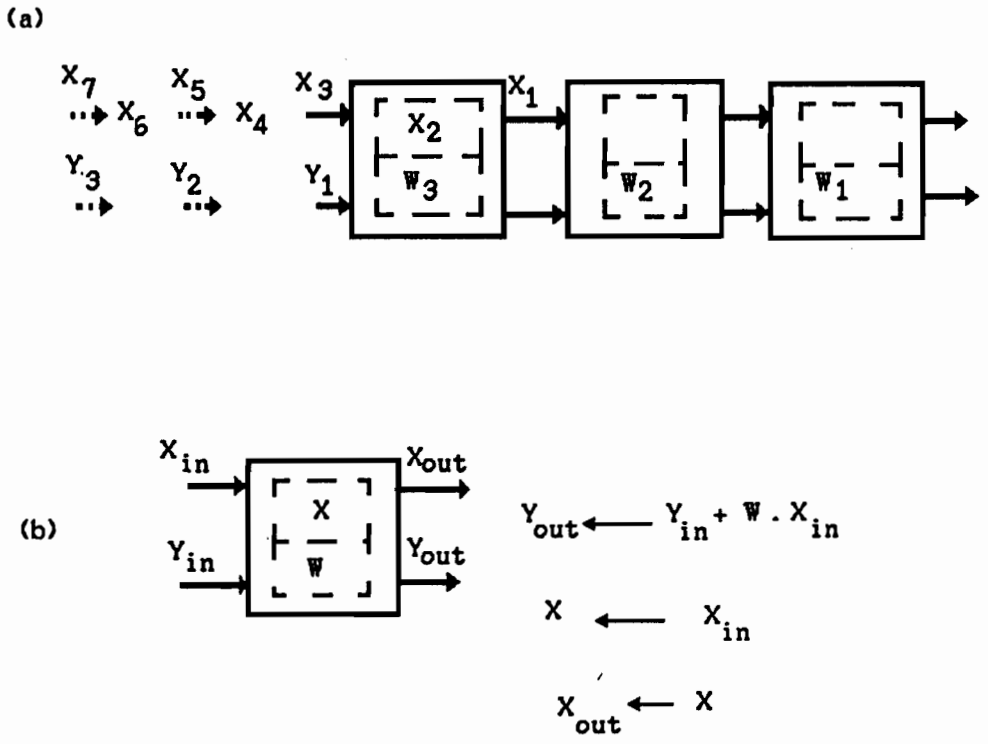


Figure 3.8 (a) Systolic convolution array and (b) Cell, where w_i's stay, x_i's and y_i's both move systolically in the same direction but at different speeds.

The designs discussed above by no means exhaust all possible systolic designs for the convolution problem. For example, it is possible to have systolic designs where results, weights and inputs all move during each cycle. It could also be advantageous to include inside each cell a "memory" capable of storing a set of weights. With this feature, using a "systolic address (or control) path" weights can be selected on-the-fly to implement interpolation, signal resampling or adaptive filtering[50]. The ESL Systolic processor[97] utilizes cell memories to implement multiple functions including convolution and matrix multiplication.

The main challenge is to understand precisely the strengths and drawbacks of each design so that an appropriate design can be selected for a given environment.

4. ARCHITECTURE OF THE CONVOLUTION PROCESSOR

The architecture selected to implement the discrete-time, two-dimensional convolution is briefly described in this chapter. Two basic principles are used, i.e., systolic arrays (as described in chapter three) and bit-serial computing. The chapter is divided into five sections. The first section gives a brief introduction of the basic requirements of the processor. The second section describes the basic VLSI systolic convolver cell. Section three gives the main architecture of the VLSI convolution processor integrated circuit (IC) containing three systolic cells. Section four gives an example of linear convolution for a systolic array to explain its operation. Finally, section five summarizes the implementation of the VLSI convolution processor.

4.1 Introduction

In order to incorporate the designed convolver into the robot workstation, it was necessary to make it compatible with the standards used in our laboratory. For example, it must be able to process images of 512x512 pixels of 256 gray levels (8-bits). Images are obtained using a black and white camera and a frame grabber that digitalizes each pixel into 256 gray scale levels. Starting from the top of the image, rows are stored one after the other in the frame grabber. Adjacent pixels in a row occupy consecutive bytes in memory. The present image resolution used is 512x512 pixels. However, the processor board (containing 27 cells or 9 IC's) can easily be programmed to handle arbitrary image sizes.

For robotic purposes, a 9x9 window size convolution gives good image processing results. The convolution involves replacing each original pixel intensity by the sum of eighty one products which are obtained by multiplying original pixel intensities by constant coefficients.[9] The original pixel intensities used are those found within a square window of 9x9 pixels on

the image while the pixel in the upper left corner of the square is the one being replaced. Multiplying eighty one by the amount of pixels found on a 512x512 picture frame results in over twenty one million multiplications and additions to convolve a single image. For this reason, if an image is to be processed rapidly, several additions and multiplications should be performed concurrently and pixels should not be repetitively read from memory.

4.2 Basic VLSI Systolic Convolver Cell

The above concurrency and minimal memory transfer specifications cannot be satisfied by computers based on the Von Newman architecture. However, processors based on a systolic architecture are easily adapted for this task. A basic systolic cell designed in our VLSI laboratory is the building block of the designed convolution card. Other commercial versions are now available[61],[54]. Figure 4.1 outlines the basic arrangement of the systolic cell. Figure 4.2 gives the block diagram with all control signals included. It contains three shift registers that can be loaded serially: the intensity (X), the coefficient (C) and the partial sum registers (Y). The overflow register (V) will be described in section 5.3. The cell's only function is to multiply the 16 bit content of the intensity register with the 8 bit content of the coefficient register and add this product to the 16 bit content of the partial sum register. The control signal functions are summarized in table 4.1.

4.3 VLSI Convolution Processor Architecture

The designed processor[44] is a 40-pin dual in-line package (DIP). The package has a removable lid which permits physical access to the chip and bonding wires, for example, to allow for probing or use of other diagnostic procedures. The design was implemented using the CMOS3 DLM[40], version of VLSI process technology which was available to our VLSI laboratory. In this technology a double well, double metal CMOS process with a 3-micron

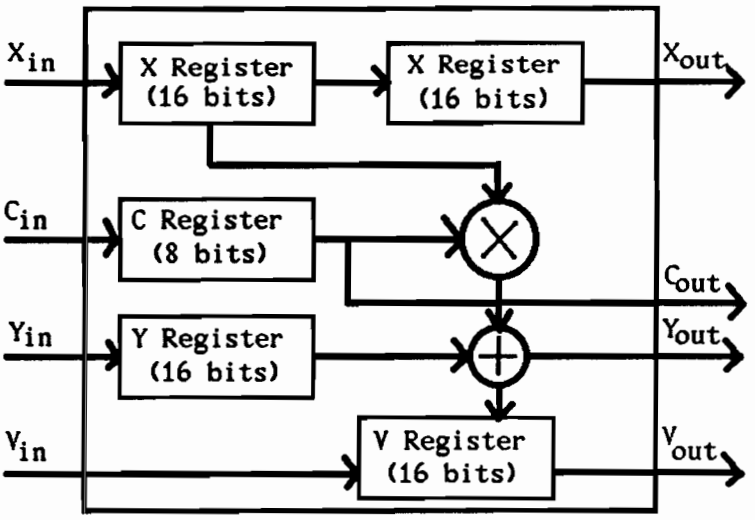


Figure 4.1 VLSI Systolic Cell

minimum feature size was employed. The Northern Telecom CMOS3 DLM process is a single polysilicon, double metal process and has additional implant and doping steps when compared to the old versions of technologies such as CMOS1B. The reduced dimensions in the CMOS3 DLM process limits the supply voltages of the finished devices to 5 volts. More recently, the CMOS1.2 DLM technology has become available in our laboratory.

The processor contains three basic VLSI convolver cells. Two cells are internally cascaded and the third cell can be externally cascaded when desired. This allows one to use the processor as a one-dimensional (1-D) filter for signal processing applications, or as a two-dimensional (2-D) filter for image processing applications such as convolution in this case. The window and/or precision of a 1-D or 2-D filter is expandable using more processors with minimal external logic. The processor is ideally suited for **"video-rate"** image processing such as template matching, noise removal or inverse filtering. Two dimensional image processing such as edge enhancement, and edge detection can be carried out fast enough for **"real time"** robotic applications. The processor block diagram given in Figure 4.3 shows the implementation of the three systollic cells in a single IC. The bonding diagram of the chip

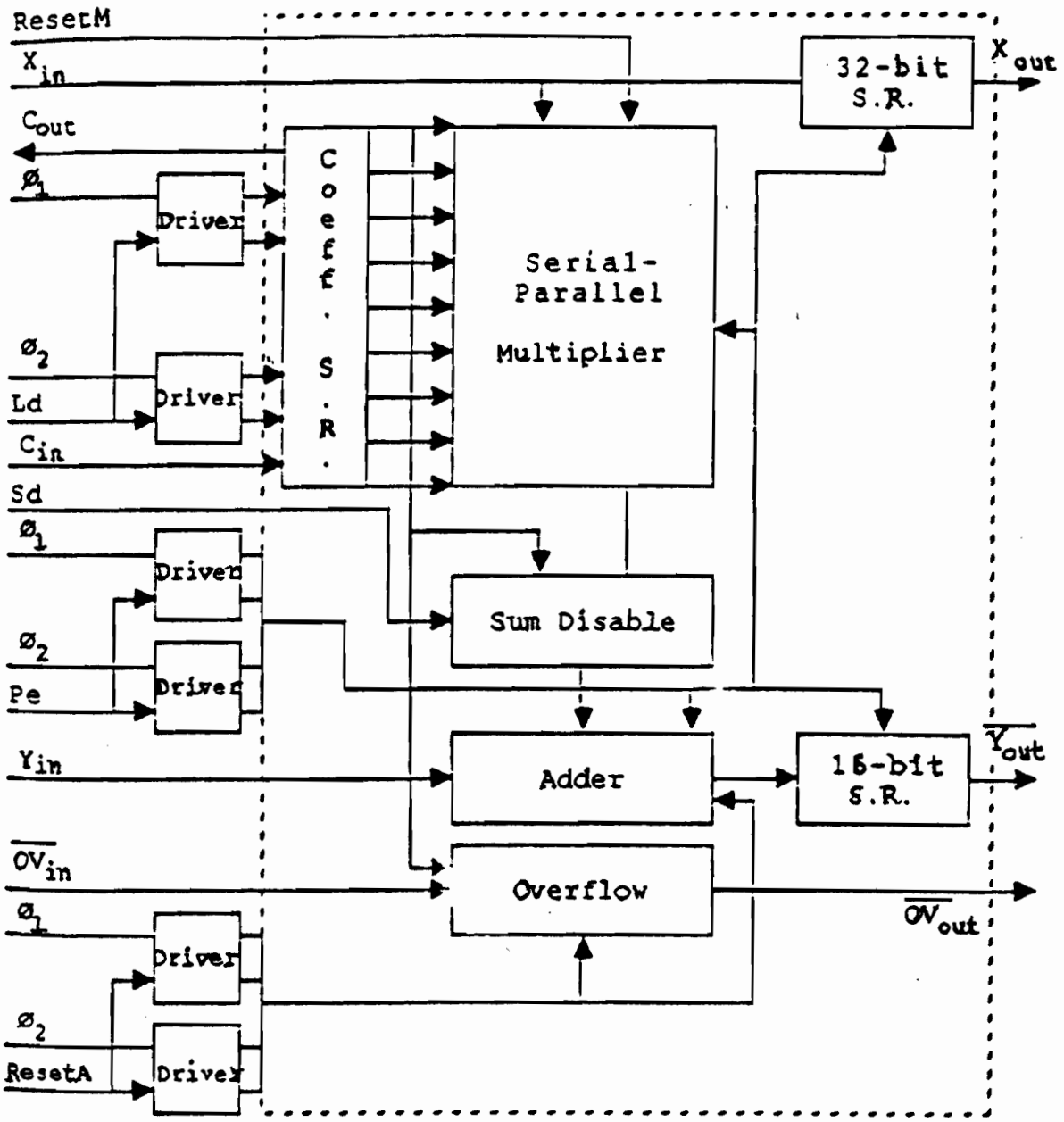


Figure 4.2 Processor Architecture Block Diagram

after fabrication is as given in Figure 4.4 and the equivalent pin assignment is as shown in Table 4.2. Finally, the complete chip plot or layout is given in Appendix E and the corresponding SDA schematics in Appendix A.

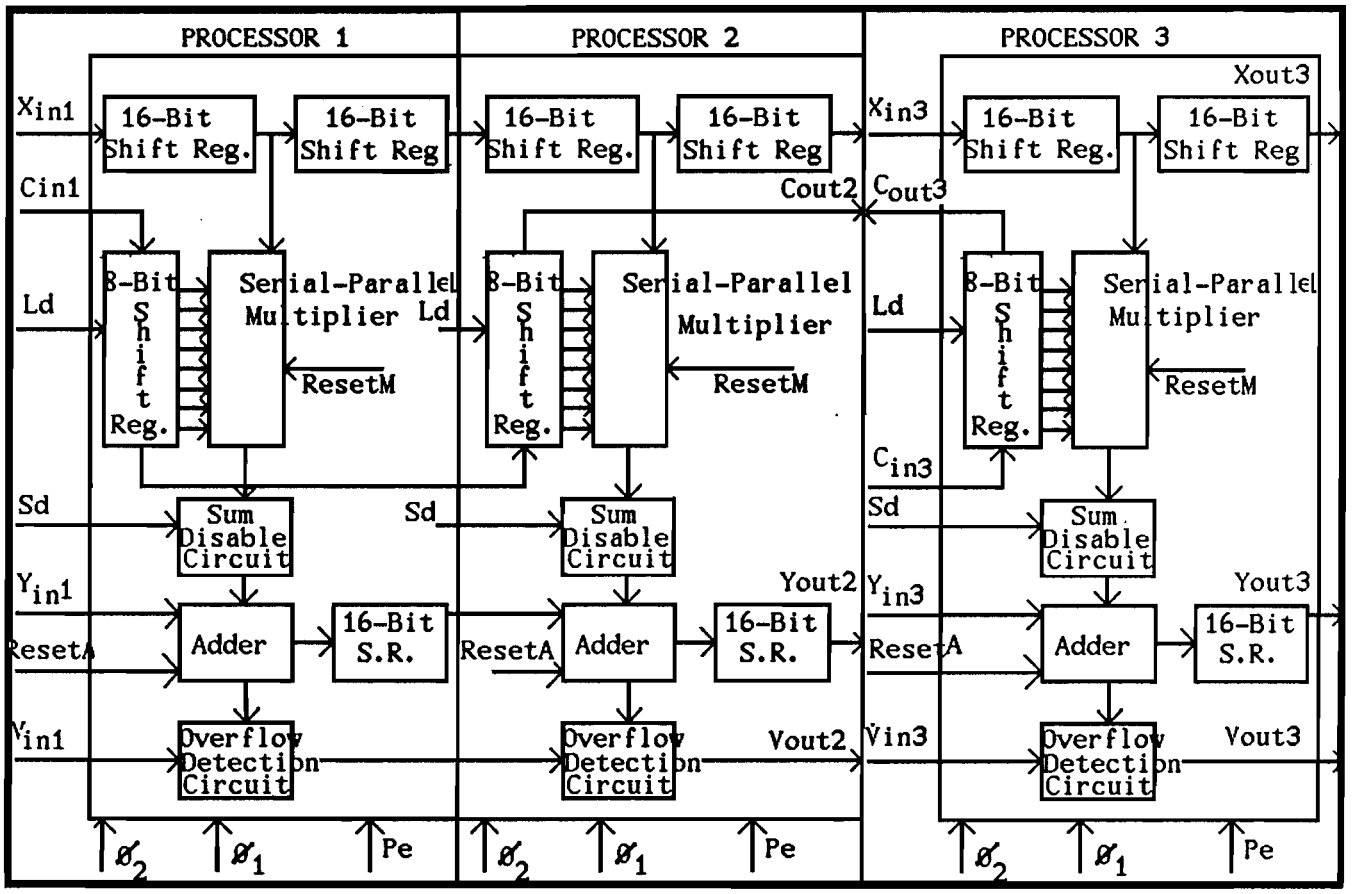


Figure 4.3 Convolution Processor Block Diagram

Signal	Description
X_in	Input signal, the pixel grey level
C_in	Input signal, convolution coefficient
Y_in	Input signal, partial convolution sum
OV_in	Input signal, the overflow detected
X_out	Output signal, delayed X_in signal
C_out	Output signal, convolution coefficient
Y_out	Output signal, the partial sum computed
OV_out	Output signal, overflow flag for Y_out
Ld	Control signal, activates coefficient loading
Pe	Control signal, activates the processor
$ResetA$	Control signal, resets the adder circuit
$ResetM$	Control signal, resets the multiplier
Sd	Control signal, disables multiplier output
ϕ_1, ϕ_2	two-phase, non-overlapping clocks

Table 4.1. Convolution Processor Signal Description

PACKAGE		LID					
	IDK40F1-192G		C-493-175-35M				
WIRE ALLOY	99% Al/1% Si	DIA.	.001"	ELONG.	1.5 - 4%	T.S.	14-16 gms
	99% Al/1% Si		.00125"		1.5 - 4%		18-22 gms
D/A PREFORM	ALLOY 98% Au/2% Si	RECOMMENDED SIZE		W/B METHOD	U.S.		

BONDING DIAGRAM NOTES: 1. DIE ATTACH PAD SIZE: .400 X .400

2. ZERO GROUND

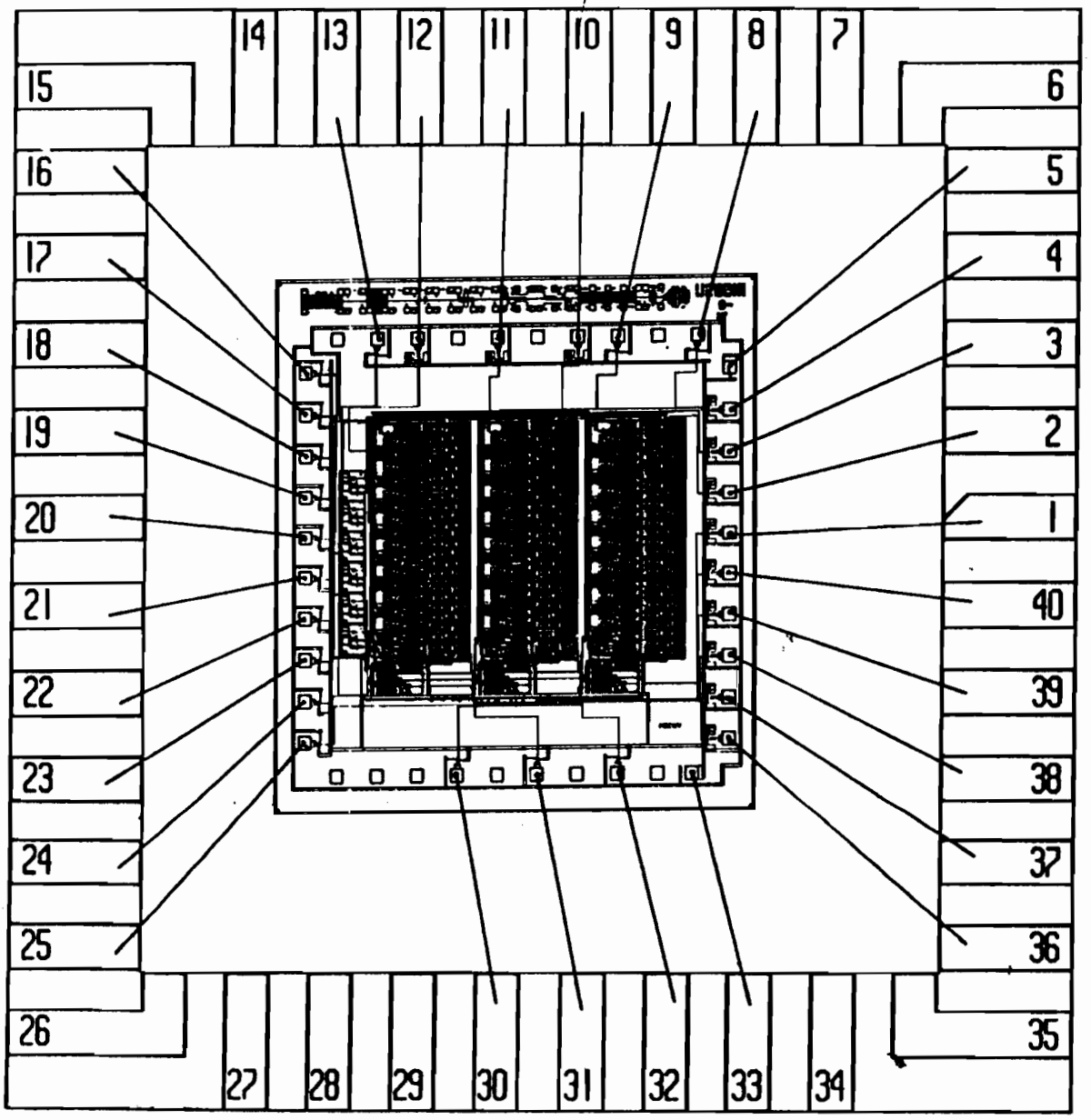


Figure 4.4 Convolution Processor Bonding Diagram

Signal	Chip pin number
X_{in}^1	13
C_{in}^1	22
Y_{in}^1	21
OV_{in}^1	24
C_{in}^2	31
X_{in}^3	9
C_{in}^3	32
Y_{in}^3	30
OV_{in}^3	25
X_{out}^1	11
C_{out}^1	12
Y_{out}^1	37
OV_{out}^1	39
X_{out}^2	10
C_{out}^2	4
Y_{out}^2	36
OV_{out}^2	38
X_{out}^3	2
C_{out}^3	3
Y_{out}^3	1
OV_{out}^3	40
Ld	19
Pe	16
$ResetA$	20
$ResetM$	8
Sd	23
ϕ_1	17
ϕ_2	18
Vdd	5
Vss	33
Not connected=nc	6,7,14,15,26,27,28,29,34,35

Table 4.2. Pin Assignment of the Convolution Processor

4.4 Operation of a Systolic Array

Figure 4.5 shows an example of the operation of a one dimensional convolution array based on the systolic cell shown in Figure 4.1. The two word delays (32 clock-cycles) in the input signal path, and the single delay (16 clock-cycles) in the partial-sum path can clearly be seen in Figure 4.1. The coefficient pins are omitted in Figure 4.5 since the convolution coefficients a, b, and c are assumed to be preloaded inside each cell for simplicity.

Let $x(t)$ represent the input sequence and a, b and c represent the convolution coefficients. The example shows the status of the linear array every 16 clock cycles on word boundaries. When data is undefined in the array, it is shown blank. Because of pipeline initialization requirements of the array, in this case, valid outputs are on every word cycle starting with t=80 (word 5).

The operation is relatively simple. Every 16-cycles, intensities move through the shift-registers, advancing through one processor every 32 clock-cycles, while the partial results of the convolution, circulating on the "y" inputs, move forward through one processor every 16 clock-cycles. Because the partial result moves twice as fast through the array, this allows it to "catch up" with the input signal already in the array. Hence, the necessary data and partial result can meet at the appropriate location for their combination. For example at t=32, the leftmost cell computes its partial sum y(0)=cx(1) shown as an output to the middle cell. At t=80, the rightmost cell generates the output sum y(1)=cx(2)+bx(1)+ax(0) which constitutes the desired convolution output result for the first image pixel. The operation of the overflow mechanism[9],[24] is described in Chapter 5. The functional description of the convolver when incorporated in the convolution card will be presented in Chapter 6.

One important characteristic of systolic arrays is the expandability they offer. They

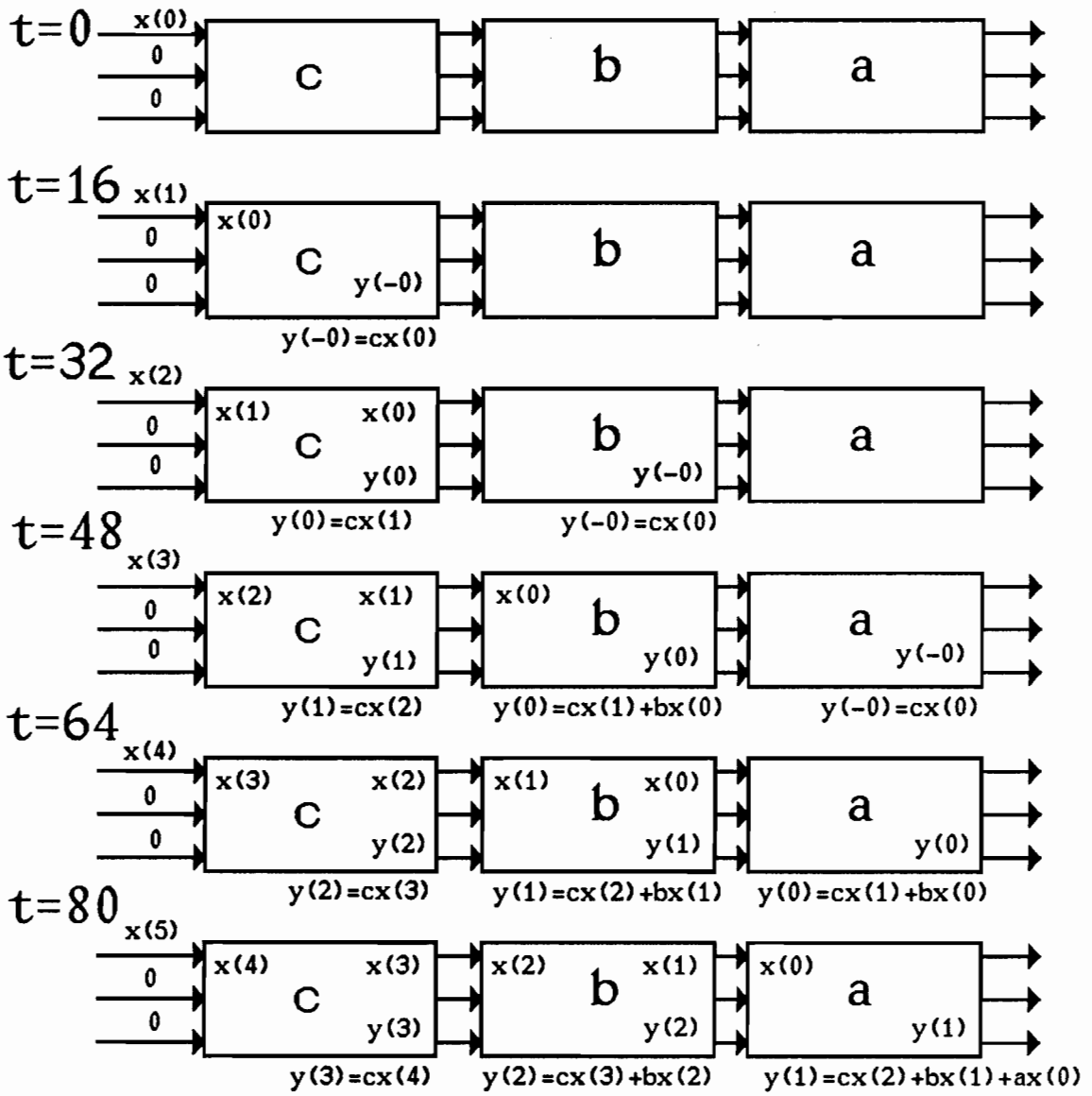


Figure 4.5 Linear Convolution Example

enable the user to employ the number of processors required for the particular application. If an array of more than 11x11 is required, it is possible to add more processors to handle the increased requirements, without any loss in performance. Grouping more processors increases the kernel size, but the augmented system still appears with the same I/O requirements and structure.

Modification of the circuit is relatively simple to handle the two-dimensional signals. A

number of 1-D convolvers are used in parallel, each operating on a different line, while their outputs are summed to obtain the desired output result. Line delay structures are needed at the inputs to properly synchronize the row data.

4.5 Implementation of a VLSI Convolution Chip

Details of the design and operation of the different parts of an initial version of the processor are briefly described by Boudreault[9],[10]. The design was implemented through an agreement between the Northern Telecom and the Canadian Microelectronics Corporation (CMC) using the CMOS5 technology to realize two systolic cells on a single integrated circuit. It was redesigned using the CMOS3 DLM[40], technology to incorporate three systolic cells on one chip or IC.

One approach for implementing CMOS circuits is to use static gates, which are very similar to normal digital gates in that they are not clocked. The advantages of a static implementation include the relatively low power consumption and the small number of transistors switching simultaneously. However, the disadvantage is the larger area required for static CMOS gates, since the logic must be duplicated for both the n- and p-type transistors. To avoid this duplication process an alternate approach using dynamic CMOS gates is often used. Many design techniques have been devised to reduce dynamic gate problems such as substrate latch-up[93], domino logic[48],[38], etc.

Static CMOS gates were selected for this project because they are relatively easy to use and because they reduce chances of substrate latch-up problems. The use of static gates was acceptable because of the silicon area available for this design.

The following techniques were used to fulfill the requirements of the convolution processor. The design uses multi-processor techniques and pipelining to divide the computational

task among a number of identical processors. An N-point convolution can be realized using N of the processors operating in pipeline mode. The processors may be used to compute any linear, shift-invariant filters and correlation functions, since these mathematical equations are very similar[81]. The capability of supporting arbitrary length of convolvers and of creating two-dimensional convolution arrays from one-dimensional convolvers were incorporated in the design specifications.

Bit-serial communication techniques were also used throughout the design to reduce the number of I/O pins required for each chip. Because this allows the use of smaller packages, more processors may be put into a specific area. It also reduces the number of printed circuit-board traces required, hence reducing manufacturing costs. The use of shift-registers to hold the coefficients and their availability as outputs makes it possible to chain them and reduce the board's I/O pin requirements.

Bit-serial processing was also adopted. Because the bit-serial computational, elements are much smaller than their bit-parallel equivalents, the integration of more capabilities (processors, dynamic range, etc.) was achieved. The coefficients were represented as 8-bit, signed numbers (-128 to +127), while the input signals were represented as 8-bit, unsigned numbers (0 to 255, left padded with zeroes to fill the 16 bit intensity register). This allowed easy interfacing with the existing facilities operating on 8-bit images, and giving a reasonable range of coefficient values.

Some error detection was provided through the overflow detection circuit, which indicates the occurrence of errors while computing the convolution sums. Two techniques proposed by Boudreault[10] were incorporated to reduce the number of possible overflows, one involving prebiasing of the convolution sum, while the other involved the use of the sum-disabling (Sd) control signal to scale the multiplier output by an integer number of bits. Three systolic

processors were placed on a single integrated circuit of 40 mm^2, with all necessary interconnections done internally. A nominal operating clock frequency of 16 MHz was specified for the design.

5. CONVOLUTION BUS CARD ARCHITECTURE

5.1 Introduction

In this chapter the architecture of a convolution processor card based on the VLSI systolic cell and interfacing with the Intel MULTIBUS is presented. The important features of the designed board are summarized in Table 5.1.

Feature	Description
Image size	Variable picture frame size
Speed	9x9 convolution in about 1s
Input level	256 gray scale level inputs
Result precision	16 bit precision results
Control	Completely DMA controlled
Compatibility	Fully MULTIBUS compatible
Interrupt capability	Interrupt at end of DMA
Data	unsigned 8-bit input data
Configuration	Reconfigurable for 1-D and 2-D
Overflow	Overflow detection capability
Card size	Fits 1 MULTIBUS slot

Table 5.1. Features of the Processor Card

Using several of our systolic cells in the processor design exploits parallel processing of concurrent multiplications and additions. Furthermore, connecting the output of the intensity register from one cell to the input of the intensity register on the right adjacent cell enables a pixel intensity to be used several times while having been moved from memory only once. Ultimately, one could have used eighty one of these cells in this circuit however, for practical purposes of board area limitations, only 27 cells could be accomodated with the result that the 9x9 convolutions must be achieved in three passes.

The 27 systolic cells are arranged in three rows of nine as shown in Figure 5.1. All register outputs on one cell are connected to the corresponding inputs on the right adjacent cell. The output of the partial sum registers of the right most cells on the two top rows are connected to the input of the partial sum registers of the left most cells on the two bottom rows. The resulting pixel is collected at the output of the partial sum register of the right most cell on the bottom row. Careful calculation reveals that, for proper operation, pixels i, $i + row\_length$ - 9, and $i + 2(row\_length$ - 9) should be fed as inputs to the intensity register of the left most cells in rows one, two, and three respectively. The processing of the resulting convolution output O will be discussed later in Chapter 6.

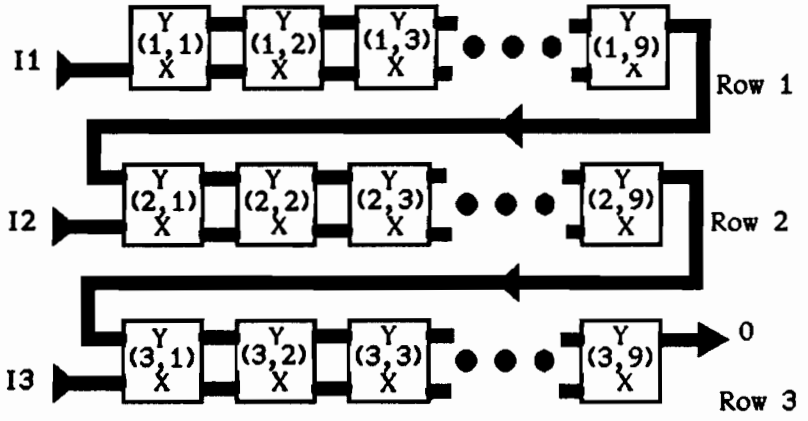


Figure 5.1 3x9 Systolic Convolution Array

Two methods can be used to supply the pixel intensities to the systolic array. The first one is to supply the intensities only to the bottom line and use appropriate shift registers to feed the same pixel intensities to rows two and one after proper delay. However for variable image sizes it becomes difficult to obtain shift registers of the required size. The other alternative which has been adopted is to use three pointers to fetch three image data rows from memory and feed the systolic array with the required data streams. This method requires each pixel to be moved three times from memory instead of once, but is easily

implementable and does not restrict the image row length to a fixed value.

5.2 Systolic Data Inter-Communications

The convolution board interterfaces directly with the MULTIBUS and the 286 CPU of the system 310 is used to sequentially load the coefficient values into the systolic array. All coefficient registers are chained into a long shift register. A parallel load shift register accepts bytes of coefficients sent by the CPU. Its serial output feeds the input of the coefficient register of the first cell in the chain. When a coefficient has been loaded into the parallel load shift register, a state machine circuit clocks every coefficient shift register in the chain for eight cycles. Transferring each coefficient in reverse order ensures that, when all twenty seven have been transferred, each coefficient occupies its desired systolic cell. The output of the coefficient register of the last cell in the chain is connected to the serial input of a parallel output shift register. The parallel output shift register feeds the data bus and serves for debugging purposes, allowing the reading of each coefficient as they are shifted out of the last cell in the chain. Moreover, the parallel output shift register is used to store the circuit configuration mode bit as will be explained in Chapter 6.

The System 310 CPU could have also been used to move all pixel intensities between the system memory and the systolic array. The CPU would have to load each pixel from memory into an internal register and then proceed to send it to the systolic array. This is not very efficient as it implies two bus cycle transfers. On the other hand, using a DMA controller chip to move the data is very efficient since it allows for fast one bus cycle transfers. The Intel 82258 Advanced DMA Controller[47] was chosen for this task because of its four independent high speed channels. Figure 5.2 outlines the circuit's basic architecture. In this application, channel 0 is used to transfer convolved pixels back into memory while channel 1, 2, and 3 are used to feed each line of the systolic array. Programming the operation of

the 82258 involves initializing some 50 I/O mapped registers[47]. A detailed description of the DMA subsystem is given in Chapter 6.

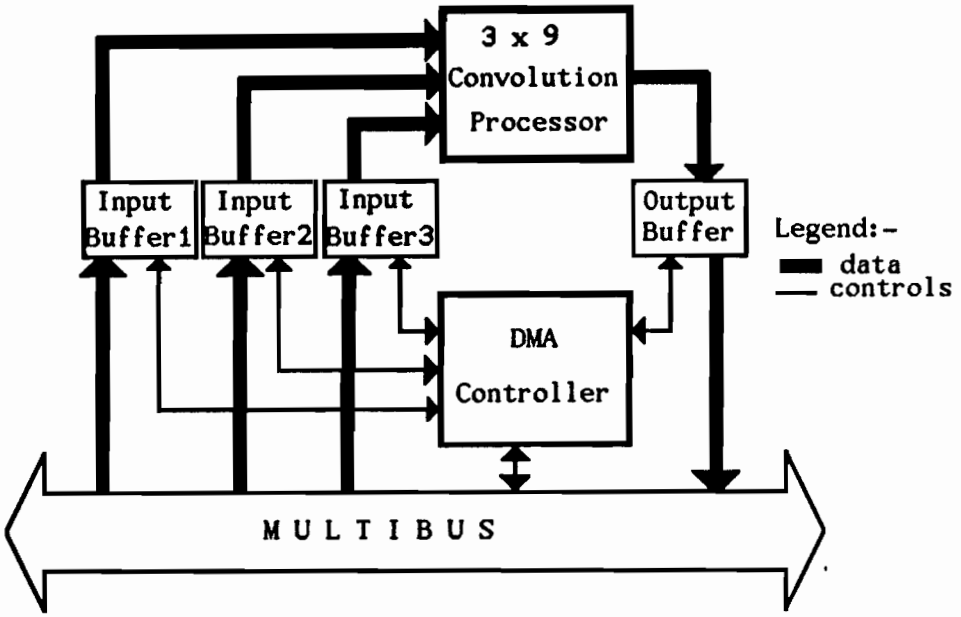


Figure 5.2 Convolution Card Architecture

The DMA can service only one channel at any point in time. For maximum system efficiency, the DMA acquisition overhead was reduced by using FIFO's to buffer each channel. Three 8-bit wide FIFO's are used for the input channels while a 16-bit width is used for the output channel. Each input FIFO has its output connected to a parallel load shift register. The serial output of each shift register is connected to one of the input (I) of the systolic array (Fig. 5.1). Six 3-state octal latches are used to interface the 16 bit input data path with the 8 bit FIFO's. Two 8-bit pixels are transferred from memory at the same time using a 16 bit word transaction on the MULTIBUS. Two octal latches are used to latch both pixels from the bus. Their 3-state outputs, permit the two bytes to be multiplexed into the same FIFO.

The output FIFO is 16 bits wide and its output buffer connects onto the data bus. The convolved pixels are 16 bits in length when they come out of the systolic array. For improved computer vision results, it might be desired to keep the 16-bit precision. On the other hand, if the resulting image is to be displayed in 256 gray scale levels, only 8 bits can be kept and depending on the chosen coefficient values, the high or low bytes are discarded. The processor hardware is built to perform the above three options. The selected mode is encoded with two bits that are stored in the parallel output shift register. When 8-bit precision is needed, two truncated pixels are concatenated into a 16-bit word and transferred to memory in one cycle as discussed in Chapter 6.

5.3 Mechanisms in Error Detection and Control

The systolic cell is equipped with an error mechanism to detect overflows. When in the 16-bit precision mode, the user can opt to have the errors reported or not. When errors are reported, a status register indicates if overflows occurred or not. Also, bit 0 of each output pixel is made to reflect whether or not an overflow was obtained at any stage in the output pixel's computation. Scanning each bit 0 on the resulting image specifies where the overflows have occurred. When the errors are not reported, bit 0 is left unchanged.

As soon as the channel 0 DMA sequence is ended, an interrupt is generated whether or not the termination was caused by an error. The faulty terminations can be identified by reading the content of the error status registers found in the DMA chip. Chapter 6 discusses how the interrupts are signaled to the CPU through the MULTIBUS by driving one of the sixteen interrupt lines with the end of channel 0 DMA signal of the 82258 DMA chip.

Considerable logic circuitry is needed to coordinate every component of the design. The systolic array must be allowed to convolve only when no input FIFO is "empty" and the

output FIFO is not "full". A combination of the DACKx signal from the 82258 and the XACK from the MULTIBUS is used to establish which FIFO should latch the data from the bus and when to do so. To minimize the chip count, all of this necessary logic is implemented using PALs. PAL20R6 and PAL20L8 series were used. The PAL designs are summarized in Appendix C which gives their logic representations and Boolean equations.

5.4 Execution and Implementation Status

To perform a complete 9x9 window size convolution, the image is convolved three times with different coefficient sets and each resulting image is stored in three different areas of memory. The final image is then obtained by adding the three partial images together. This is easily done with a fourth pass through the convolution processor, having the three left most coefficients equal to one and every other equal to zero. The DMA controller is able to chain DMA sequences without the intervention of the CPU by reading the next command directly from memory. However, in this design one can not use this feature to perform the four successive passes because the CPU must load the new set of coefficients between each pass. To simplify the hardware requirements in implementing the processor design, the coefficient loading is first carried out by the CPU in the I/O space. The CPU then sets up the 82258 registers for the convolution to occur by DMA.

There are 262,144 pixels in a 512x512 image and every pixel must be fed once through each of the three input channels. Also, the same number of resulting pixels must be transferred back into memory. This means that 1,048,576 pixels must be transferred for one pass giving 4,194,304 pixel transfers for a complete 9x9 convolution in four passes. In the 8-bit output precision mode, both the input and resulting pixels are transferred in pairs resulting in approximately two million memory transfers for a complete convolution. In the 16-bit output precision mode, only one output pixel can be transferred in one cycle. Therefore,

approximately 2,500,000 memory transfers are required to completely convolve an image, an increase of twenty five per cent. The 82258 can perform 2,666,000 memory transfers in one second at 16 MHz. Each systolic cell must process 262,144 pixels during each pass or about a million pixels for a complete 9x9 convolution. The present version of the systolic cell can process one million pixels in one second at the 16 MHz nominal frequency. Therefore, the systolic array is the limiting factor responsible for the one second time required to obtain a complete convolved image.

The circuit has been simulated at the architectural level using a program written in PASCAL and running under MSDOS on an IBM PC[20]. This program simulates the control strategy used to manage the DMA transactions which service the input and output FIFO's. The contents of the FIFO's were shown dynamically on the screen using bargraphs. The overall efficiency of the design proved to be excellent. Except for the initial transient associated with filling the three input FIFOs, the systolic array is kept operating continuously throughout the entire convolution period. The operating statistics generated by the simulation program showed that approximately twenty five percent of the bus bandwidth remained available to the CPU during the convolution processing because of the periodic nature of the DMA servicing. The simulation and performance evaluation is discussed in detail in Chapter 7.

For the construction phase, the schematics, the parts and wire list were produced using ORCAD version 3.1. A program was written to convert the ORCAD IC oriented wire listing into a more suitable board coordinate oriented version. The circuit has been assembled on a 36 column Vero Speed-wire MULTIBUS bus card. The board is currently under test in an Intel System 310 running iRMX 286 real-time multitasking operating system. Image processing software for frame grabbing of TV camera images and their display on a RGB color display is being developed for the RMX environment. Current research developments

also include designing a new VLSI 32-bit floating point systolic convolution processor [25],[55].

5.5 Convolution Card Hardware Design

This section presents an overview of the hardware design of the convolution processor board. The high performance convolution bus card is designed to directly interface the Intel MULTIBUS. It takes an image stored in the memory of the host computer, performs a desired convolution and stores the resulting picture back in memory.

The Intel 82258 ADMA (Advanced Direct Memory Access) Coprocessor chip was selected for the DMA control because of its MULTIBUS I compatibility characterized by 24 address lines, 16 data lines, 12 control lines, 9 interrupt lines and 6 bus exchange lines. It has 4 independently programmable DMA channels. Hence, we can program three channels to transfer source image data to the input FIFOs and the fourth channel is programmed to transfer the convolved pixel data from the output FIFO to memory on the MULTIBUS. These channels function with a simple DMA request (DREQ) and DMA acknowledge (DACK#) protocol. Here a DMA transfer from the source image to the input FIFO will be requested when an input FIFO becomes "empty" or conversely when the output FIFO becomes "full" and needs to be drained.

The ADMA starts up in the 286 Mode after reset. It is subsequently programmed to operate in the Remote mode where the ADMA is meant to be the sole local or resident bus master interfacing to the system processor through the 82289 bus arbiter. A complete description of the capabilities of the 82258 ADMA Controller can be found in the user's guide[47].

The 82258 ADMA controller is aided by the 82289 Bus Arbiter and the 82288 Bus

Controller in performing the necessary bus cycle transfers from and to the memory of the host computer. When signaled by the 82258, the input FIFOs grab the latched data from the bus. Similarly, the output FIFO data is transimitted directly to the bus which allows for fast single cycle bus transfers. The block diagram of Figure 5.3 shows the three different channels used to input the image through the systolic convolution array. Because the image format is made up of 8-bit pixels, it is possible to transfer two bytes at the same time when doing a word transfer. This multiplexing is achieved using PAL7 given in Appendix C and is shown schematically in sheet 4 of Appendix B. The two latches used to buffer and multiplex the internal data bus are regulated by the *MSBEN* output of PAL7.

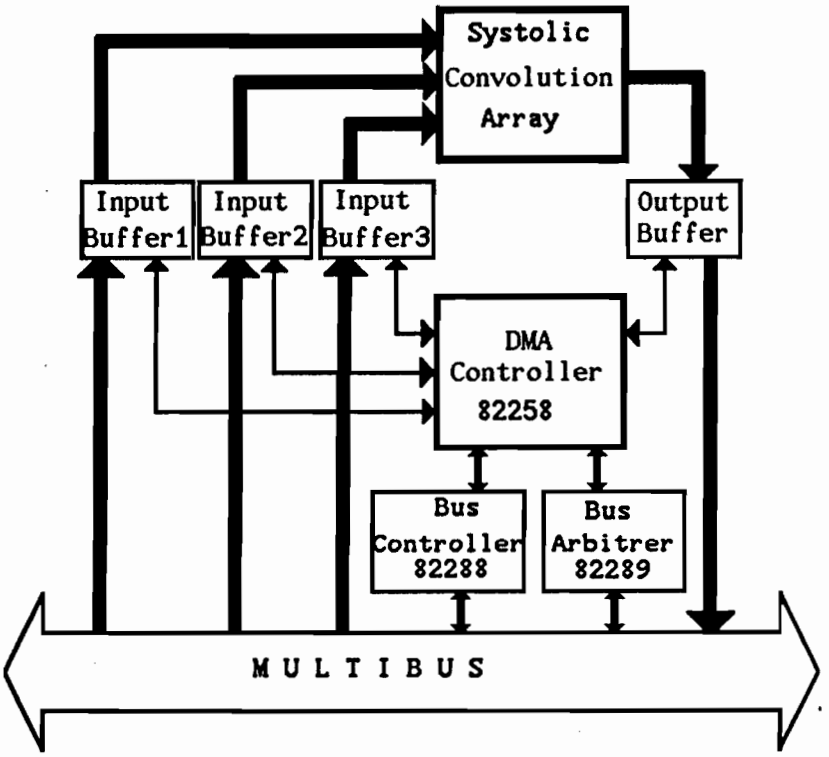


Figure 5.3 System Design Block Diagram

Every 16 clock cycles of the convolver, three new bytes must be loaded into the shift

registers that feed the inputs of the convolver. Design sheet 5 in Appendix B shows the 3 pairs of FIFO's used as input buffers. The 3 shift registers (SFTIN1, SFTIN2 and SFTIN3) have pin 18 grounded so that 0's will be shifted for the eight last clock cycles. The convolver input format requires the pixel intensities as the low bytes padded with 0's to form a 16-bit word. The shift register controls are derived from PAL1 as given in Appendix C.

Design sheet 10 of Appendix B shows the convolver array (3 rows of 9 processors) with its 3 inputs (CIN, XIN, and YIN) as stated in Table 4.1. The CIN and LD signals are used to input the coefficient values into the convolver while the image intensities XIN, the partial sums YIN, and the overflow flags OVIN are processed by the RESETA and RESETM signals synchronized with CLK1 and CLK2. The output of the convolver is a 16 bit word representing an integer. Depending on the application, it may be desired to keep only the top 8 bits, the bottom 8 bits or to keep them all. The Qa' and Qh' signals (HoRL, 16o8) of "Coef-Out" shift register on sheet 8 of Appendix B are used to select the mode of operation. PAL2 generates the necessary controls for the operation of the convolver array as given in Appendix C. The results are always stored back into the memory of the host computer using word transfers. The 4 FIFO's on sheet 6 of Appendix B are used as a buffer between the convolver outputs and the data bus. Further details of the control signals generated by the seven PALs are presented in Appendix C giving their respective logic descriptions, associated Boolean equations and additional brief comments.

6. THE DMA SUBSYSTEM

6.1 General Description

In this chapter, the detailed operation of the DMA Controller Subsystem of Figure 5.3 will be presented. The DMA subsystem is based on the Intel ADMA 82258 chip. Because of its programmability, it permits a great flexibility in the definition of the image frame to be convolved. However, since it has only four independent data channels, the convolution window is limited to three rows since 3 channels are used for input data and 1 channel for output data. In order to accomodate operating the 82258 and the convolver subsystems at their maximum operating frequencies, these have been isolated from one another by introducing FIFOs in the data channels to exchange data between both subsystems. Thus the DMA always loads the three input FIFOs and empties the output FIFO whereas the Convolver always empties the input FIFOs and fills the output FIFO. The image to be convolved is passed in each of the three input FIFOs coming out as a convolved picture from the output FIFO.

Since the FIFO depth is only 64 units, all the 512x512 image pixels cannot be loaded in one block. The DMA must load the image into the FIFO in small packets. In addition, the DMA must do so while insuring that the Convolver will stop the least amount of times due to lack of inputs or because the output FIFO is full. To solve this problem, a simple algorithm was devised and tested by a simulation as presented in Chapter 7. Each FIFO communicates its need to be serviced to the DMA when only 8 units (almost empty) remain in an input FIFO or 56 units (almost full) are present in the output FIFO. The DMA will then choose according to the priority of the request (Output first) and start servicing that particular FIFO. The DMA will stop when the input FIFOs are almost full (i.e. 56 units) or

when the output FIFO is almost empty (i.e. only 8 units remaining). Then the DMA will service any other present request from the other FIFOs. Because of this, the image to be convolved must be augmented by 64 units to ensure that the output FIFO will be serviced often enough to give back an entire convolved picture. Otherwise, the convolver might stop operating due to lack of inputs while some of the desired outputs still remain in the output FIFO.

Thus, the DMA subsystem must perform three major functions. First it must be MASTER of the MULTIBUS for data exchange. Second it must be a SLAVE to the MULTIBUS to permit the CPU to program it by loading registers and third it must implement in hardware the servicing algorithm for the FIFOs.

6.2 ADMA as MASTER Interface

To meet the above requirements of the MULTIBUS, buffers are used as shown in Figure 6.1. The address bus latches of the ADMA are tri-stated to remove them from the bus when it is no longer its MASTER. Because the ADMA must read and write data to the MULTIBUS, transceivers are used.

The ADMA does not control the MULTIBUS directly. The 82288 Bus Controller and the 82289 Bus Arbiter translate the commands of the ADMA into their proper MULTIBUS format and timing. The Bus Arbiter is responsible for acquiring or relinquishing the bus for the ADMA. The Bus Controller generates all the read and write commands as well as some latch controls for proper synchronization.

When the ADMA needs to execute a MULTIBUS command, it first tells the Bus Arbiter that it wants to be MASTER of the bus by asserting its HOLD signal resulting in the BUSREQ# and BPRN# signals to the MULTIBUS. When the Bus Arbiter finally obtains

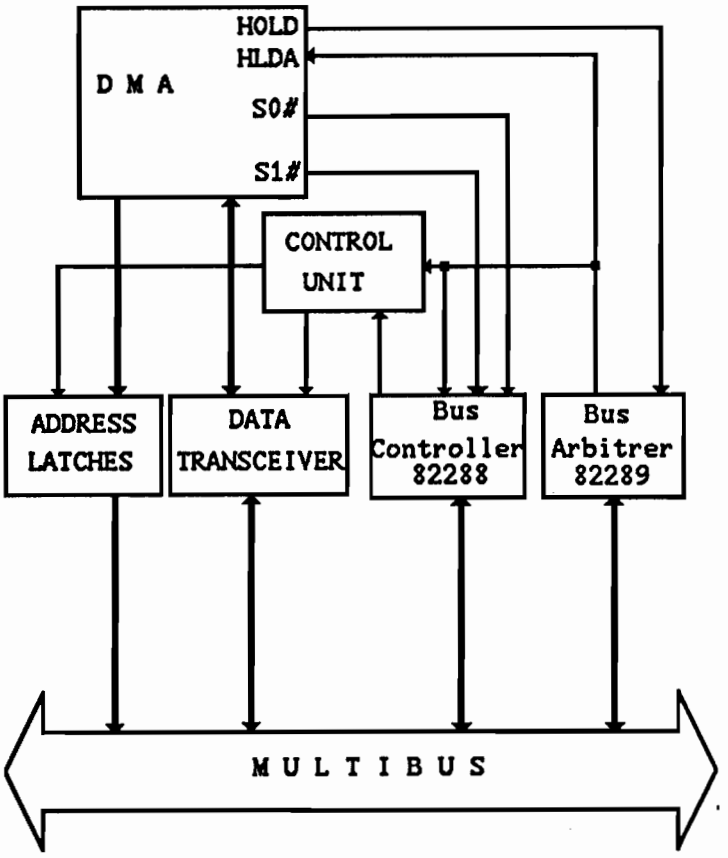


Figure 6.1 DMA Master Interface.

the bus, it activates Address Enable (AEN#) signal corresponding to the Hold Acknowledge (HLDA) signal for the 82258. The DMA subsystem is now the MASTER of the bus. The Bus Arbiter will relinquish the bus only after the last command of a block move has been completed, (HOLD and BREQ are removed).

AEN# enables the Bus Controller. When the DMA wants to read or write, it asserts S0# or S1#. This is translated by the Bus Controller into MRDC# or MWRC# at the proper time as well as the signals for synchronization of the buffers with the command. These signals are Address Latch Enable (ALE), Direction of Transceiver (DT/R#) and Data Enable (DEN). ALE loads the address latches with the desired address value. This insures address stability. DT/R# gives the direction of the data transfer (LOW for the

direction from the bus to the DMA system). DEN enables the data transceivers. Figure 6.2 shows the timing waveforms generated by the bus controller for two successive read cycles where the timing symbols are defined in Table 6.1. The read and write cycles timing are presented in Figure 6.2 and Figure 6.3 respectively.

Symbols	Parameter	Min	Max	Unit
t_1	258 Output Valid Delay	1	40	ns
t_2	ALE Active Delay from CLK	3	22	ns
t_3	ALE Inactive Delay from CLK		20	ns
t_4	DT/R# LOW from CLKck		25	ns
t_5	DT/R# HIGH from DEN Inactive	5	35	ns
t_6	DEN (read) Active from DT/R#	5	35	ns
t_7	DEN (read) Inactive from CLK	3	35	ns
t_8	CMD# Active Delay from CLK	3	25	ns
t_9	CMD# Inactive Delay from CLK	5	25	ns
t_{10}	READY# Setup Time	38		ns
t_{11}	READY# Hold Time	25		ns
t_{12}	DEN (write) Active from CLK		30	ns
t_{13}	DEN (write) Inactive from CLK	3	30	ns
t_{14}	XACK# Signal Memory Delay	..	..	ns
t_{15}	XACK# Memory Removal from CMD#		65	ns
t_{16}	READY# Inactive Delay	5		ns
t_{17}	READY# Active Delay	0	24	ns
t_{18}	Enable Active Delay from CLK*	125	125	ns
t_{19}	ShiftIn Delay from CLK**	62.5	62.5	ns
t_{20}	In/OutReady Inact.from Shin/Out		28	ns
t_{21}	In/OutReady Act. from Shin/Out		25	ns
t_{22}	In Data Stable Setup Time	3		ns
t_{23}	In Data Stable Hold Time	25		ns
t_{24}	Previous Word Stable from CLK10			ns
t_{25}	Next Word Stable from OutReady		20	ns
t_{26}	Next Word Stable from CLK		40	ns

Table 6.1 Timing Symbol Definition.

The 82284 Clock Generator and Ready Interface is used. In addition to generating the CLK and RESET signals for the entire DMA subsystem, the 82284 also synchronizes

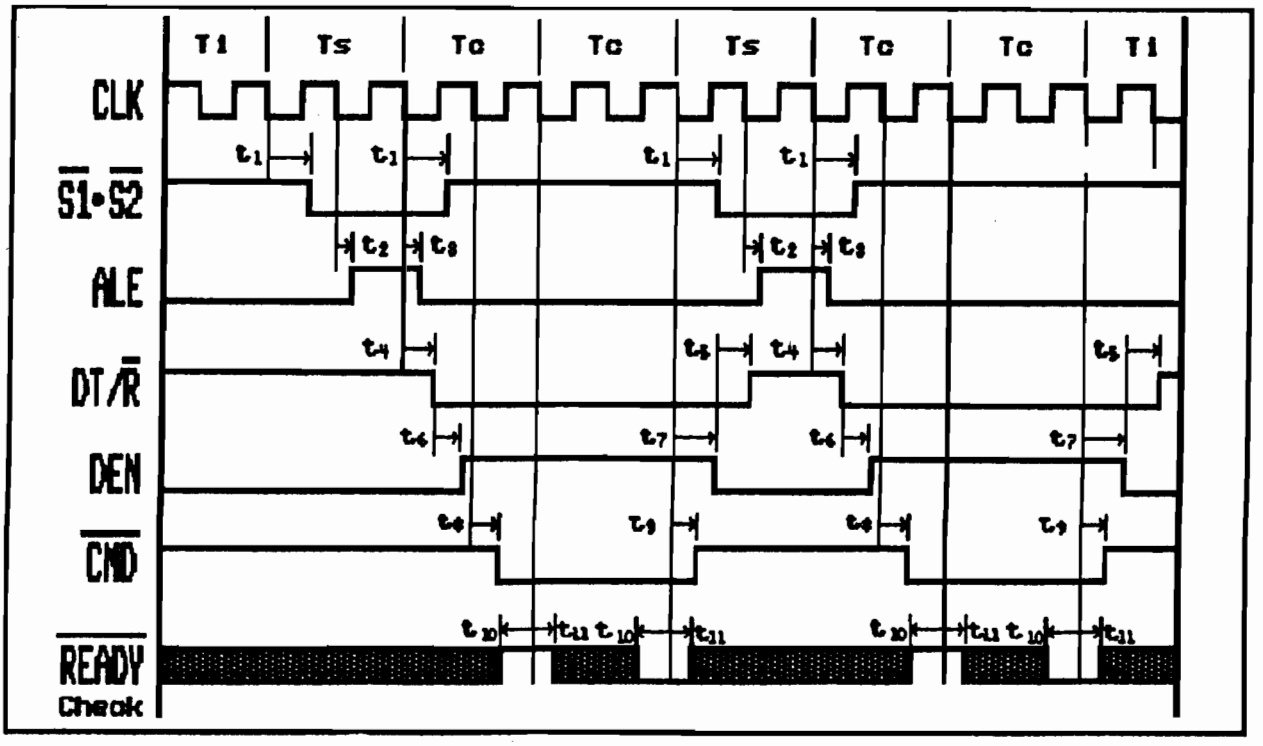


Figure 6.2 Read Cycle Signals between DMA and 82288.

the XACK# signal to form the READY# signal which regulates the ADMA operation by inserting wait states as required (see sheet 3 of Appendix B).

6.3 ADMA as SLAVE Interface

The DMA has internal registers which control the memory transfer to be executed. This implies that the CPU will initialize these registers using I/O instructions and thus must be able to access the DMA as a SLAVE as shown in Figure 6.4.

When the CPU is MASTER of the MULTIBUS, it asserts the selected address of the ADMA register. Because the internal registers of the ADMA take up the entire address space given by the lower 8 bits, 256 addresses must be reserved to access the DMA. The higher 8-bit address is recognized by the Address Decoder which generates the Chip Select

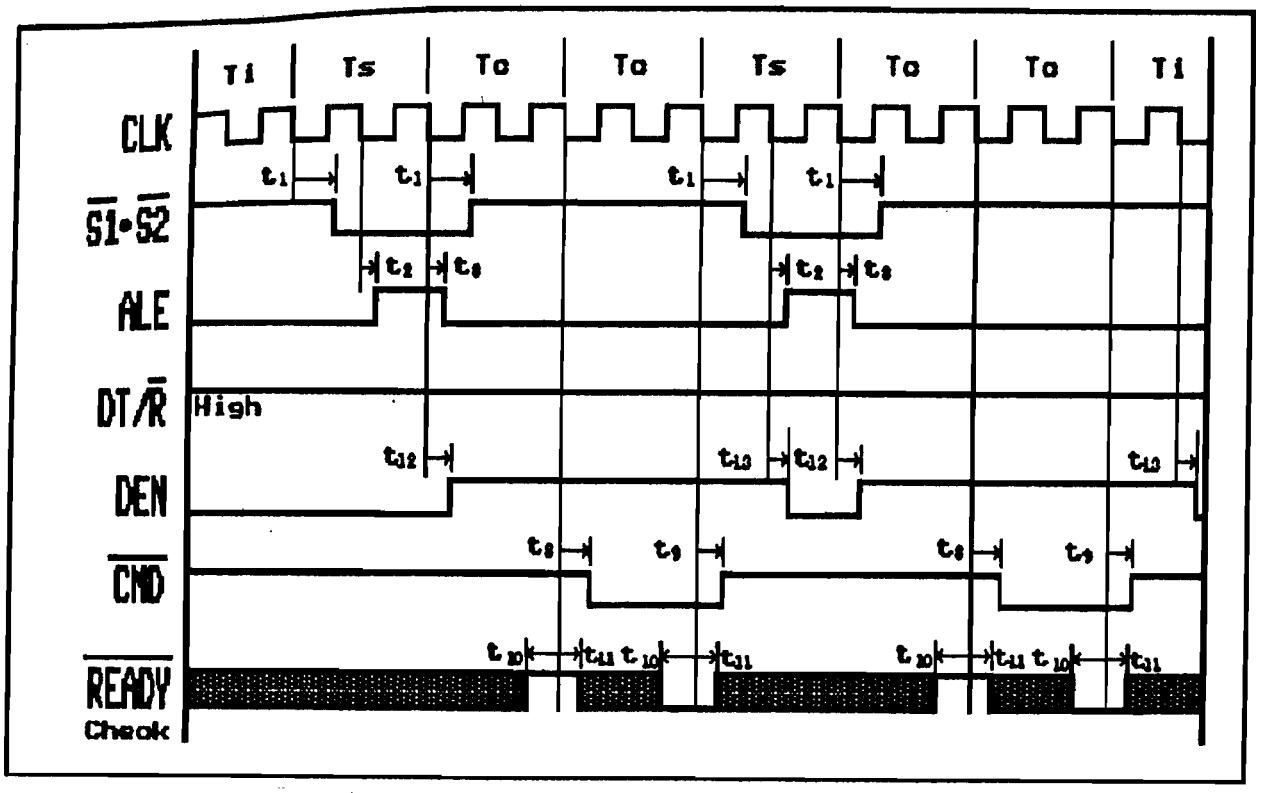


Figure 6.3 Write Cycle Signals between DMA and 82288.

(CS#) signal to the DMA. Upon receiving this signal, the DMA operating in Remote Mode sends back the (local) Bus Relinquish (BREL) signal which is used to control the address drivers and data transceivers, through the Control Unit. With WR# and RD# wired to the IOWC# and IORC# lines respectively, the DMA registers can be accessed by the CPU. A special counter in PAL5 generates the XACK# signal after reception of a bus command. The delay introduced by the counter is larger than the worst time taken by the DMA to access its registers. The DMA ceases to be a SLAVE as soon as CS# is de-activated.

6.4 ADMA to FIFO Interface

Each FIFO outputs two flag signals indicating the number of units held in storage. These signals are the Almost Full/Empty (AF/E) signal, which indicates that the FIFO contents are only 8 units away from the front or the end of the queue, and the Half Full (HF) signal,

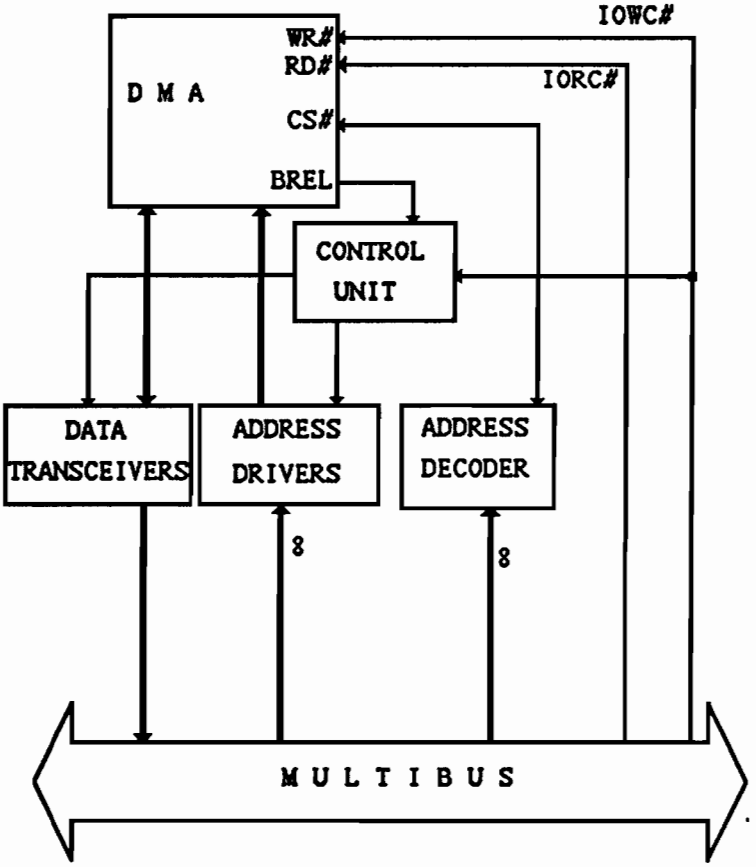


Figure 6.4 DMA as SLAVE Interface.

which signals that at least 32 units are present in the queue.

The state of each FIFO is monitored by the FIFO Control block as indicated in Figure 6.5. When one of the Input FIFOs is almost empty or if the Output FIFO is almost full, the FIFO control sends a DMA Request (DREQ#) signal corresponding to the FIFO needing to be serviced. The ADMA responds to this request only if it has finished servicing another FIFO and if no higher priority FIFO is also requesting a transfer. The priority scheme is resolved in the ADMA based on the number of the channel requesting a transfer, where channel 0 (the Output FIFO) has highest priority and channel 3 the lowest. The Output has priority over the inputs to avoid any clogging of the system. When the ADMA is ready

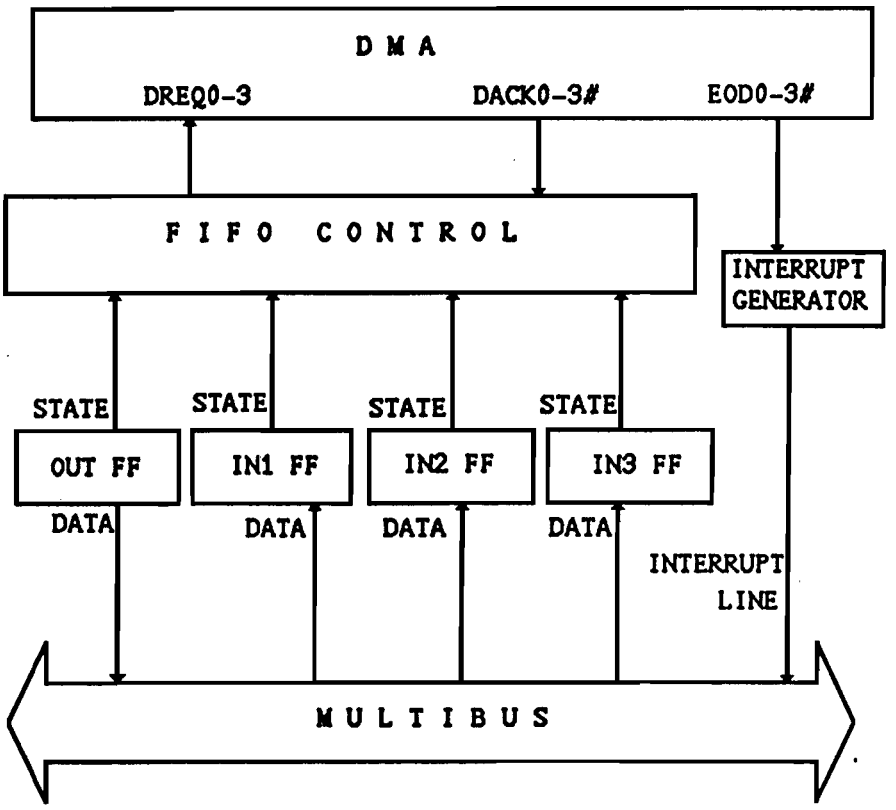


Figure 6.5 DMA as FIFO Interface.

to service a request, it asserts the corresponding DMA Acknowledge (DACK#) line.

The DACK# signal regulates the FIFO Control such that each time the ADMA reads or writes, it forces the FIFO to grab or push data on the bus at the right moment. This permits the ADMA to perform a 'single-cycle' transfer, the fastest transfer mode supported by the DMA. However, this does not mean that a single clock cycle is used. As shown in Figure 6.6 and Figure 6.7, the minimum number of clock cycles for a read or a write command is 6. When each channel has finished its program, the corresponding End of DMA (EOD#) line is activated. An interrupt to the CPU is generated when EOD0# is produced after all the outputs have been transferred.

When the DMA writes to the iRMX memory, the Control unit enables the tri-state

buffers of the output FIFO which implies that the 82258 DMA processor cannot perform 'on the fly' operations on the data. The FIFO Control sends the Shift Out (ShOut) signal to the Output FIFO before each MWRC# to output new data to the bus. In reading from the MULTIBUS memory, the DMA reads two bytes (i.e. one word) at a time resulting in a maximum reading rate of 4.7 Mbytes/second with an optimum Convolver clock frequency of 18.8 Mhz as will be discussed in Chapter 7. The FIFO Control separates each byte with the Enable (INEN) signal and loads them in turn into the requesting input FIFO using the Shift In (ShIn) signal. This is shown in Figure 6.6.

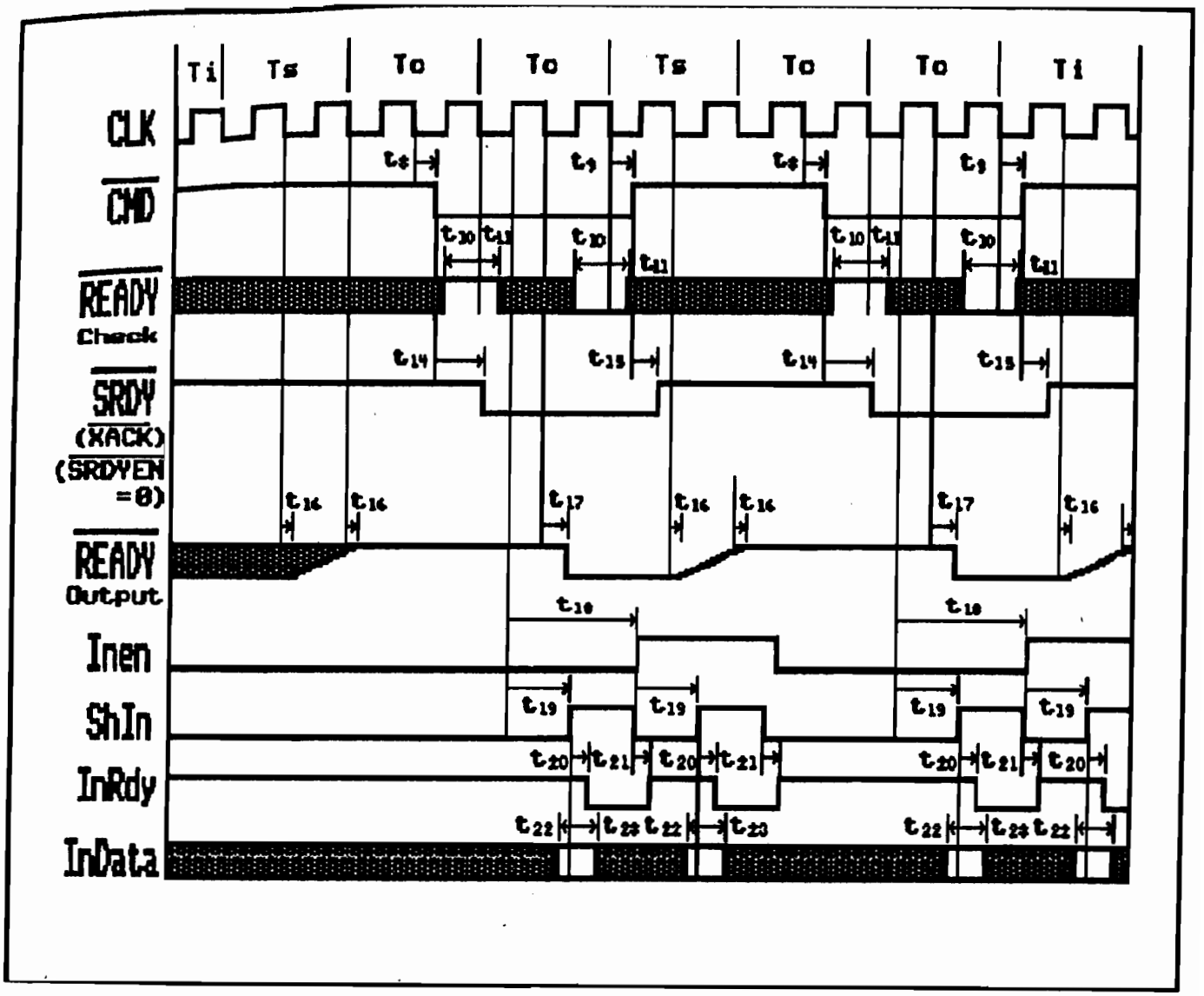


Figure 6.6 DMA Read Cycle Timing from BUS to FIFOs.

sheet 3 of Appendix B).

The crystal option on the 82284 Clock Generator was chosen requiring that the pins X1 and X2 be connected to a crystal. The CLK output gives a 50% duty cycle square wave at 16 MHz and PCLK at 8 MHz. PCLK is modified by S0# and S1# but it is not used. To ensure that the setup time for READY# at the 82258 is respected, the synchronous generation of READY# was selected. Synchronous Enable (SYEN#) is connected to AEN# so that READY# is generated only when the 82258 is MASTER of the bus. SRDY# is connected to XACK# so that READY# is produced as soon as the bus command is executed. Note that both address buffers and address latches have negated outputs because the MULTIBUS has active LOW addresses whereas the DMA has active HIGH addresses.

The data path control signals are implemented in the Control Unit by PAL5 (see sheet 2 of Appendix B). Basically it is a two channel multiplexer selected by AEN#. When AEN# is inactive (HIGH), the DMA is in potential SLAVE mode. The data transceivers are enabled (G# LOW) when BRE LOW is asserted (i.e. DMA Slave). The direction is derived from IOWC#: when HIGH (corresponding to a read command by the CPU) the transceivers are connected from the DMA to the MULTIBUS and when LOW (for a write command) the MULTIBUS is fed to the DMA.

When AEN# is active (LOW), the DMA is MASTER of the bus. The transceivers are thus only enabled when DEN# derived from 82288 is active and their direction is determined by DT/R# signal. PAL5 also contains the generation circuit of CS#. When the upper 8 bits correspond to the address defined by switch 1, the comparator outputs active LOW signal which is further qualified to ensure that the DMA will never receive an active CS# signal as long as AEN# is active. Only when the DMA is not MASTER of the bus will the result of the comparator influence CS#. Thus the internal registers of the DMA will

not interfere with the data bus during a writing operation (i.e. the servicing of the Output FIFO enabled by DACK0#). The bus tranceiver direction is regulated by the BUS/DMA# signal to transmit data to the MULTIBUS when servicing the output FIFO.

The Coef-In and Coef-Out registers are selected using one of the unused DMA register locations (address 40 HEX). PAL7 decodes this signal as SLAVE# and de-asserts CS# for this condition. When the CPU reads or writes into the DMA, the maximum delay time for the register transaction is just above 6 clock cycles. For logical simplicity, XACK# is generated by a counter (see design sheet 4 of Appendix B) which counts 8 cycles and then is locked. The counter starts counting when either IOWC# or IORC# is activated and is cleared when these commands are removed from the bus.

DREQ generation is done with PAL4 (see sheet 3 of Appendix B). It is produced for the input FIFOs when AF/E is HIGH, HF is LOW and DACK# is HIGH (i.e. the input FIFO is almost empty). When DACK# becomes LOW, DREQ remains active until both AF/E and HF are HIGH (i.e. the input FIFO is almost full). DREQ then becomes LOW until the initial condition is met again. The opposite sequence happens for DREQ0#. It is produced when both HF and AF/E are HIGH and will remain so until HF is LOW and AF/E and DACK0# are HIGH.

Shift Out generation (ShOut) is produced by a D flip-flop to give a pulse synchronized version of the ALE signal, synchronized on the rising edge of CLK and cleared as soon as DACK0# is inactive. Shift In generation for the three signals (SiIn1, SiIn2 and SiIn3) are generated by PAL6. They are started by the DCLK pulse, produced at the first rising edge of CLK with an active XACK# and force the input FIFOs to load the 2 bytes contained in the input latches. In the next chapter, the simulation study of the FIFO control strategy will be presented.

7. SIMULATION AND PERFORMANCE EVALUATION

7.1 Introduction

As the cost of building and testing prototypes increases, simulations to test a system's performance become more necessary. However, gate simulators are not the best approach when large VLSI chips are used. Too much unnecessary information is generated and they require having detailed plans. If the planned architecture does not perform correctly, the time spent designing these plans will have been wasted. This chapter summarizes how simulation at the architectural level proved to be a useful tool in the design of a high performance convolution processor card[20].

For real-time applications such as robotic vision, the convolution processor card is intended to perform a two dimensional 9x9 convolution of a 512x512 picture frame in about a second. To execute rapidly the twenty-one millions of operations involved in one convolution, this card is based on designed VLSI chips, i.e., systolic convolution cells which perform the computations in pipeline fashion, and Intel's 82258 Advanced DMA chip, which transfers the data between the host computer and the convolution cells. Still, these two subsystems require an efficient interface in order for the card to perform as desired. Architectural simulation confirmed that data queues with a simple control algorithm provide an efficient solution and enabled the design to be successfully completed.

As digital systems are required to respond to more and more diversified environments[73], they have grown more complex, often relying on expensive or specially tailored VLSI chips. Prototypes have accordingly become more expensive and greater emphasis has been put on the optimum development of these costly chips. To this end, special purpose design languages[85] and simulators have been used more than ever to test a system's performance.

The first question a simulation must answer is whether or not the global approach of the system works. This answer is not easy to obtain using conventional simulators. Due to the presence of complex VLSI chips and the systems that support them, a gate level simulator becomes cumbersome. The designer has to disentangle the answer from the mass of produced information. The use of specially tailored chips make chip level simulations[4] either difficult or necessitate programs large enough to allow the definition of new blocks. In any case, the simulation will be expensive, time-consuming, and needs the detailed description of the system. If the global approach proved to be wrong, all these efforts will have been wasted since the basic features of the design are defective.

What is needed is a simulator which will test the architecture of the system before detailed plans are drawn. If the simulator is simple enough, different alternatives can be tested rapidly and efforts can be concentrated into the most promising architecture, with good confidence that the resulting system will work.

Unfortunately, the presence of the buffers does not solve completely the problem of obtaining an efficient interface between the Convolver and the DMA. The algorithm used by the DMA to service the buffers must insure that the convolver halts for a minimum amount of time due to a lack of inputs or to a lack of place in the output FIFO. Maximum efficiency will be reached if the amount of data in the FIFO's attains a cyclical staircase configuration. The presence of pixels in all the buffers will permit the Convolver to continue to function while the DMA fills the input buffer or empties the output buffer. The cyclical motion of the staircase configuration among the buffers allows each FIFO to be serviced in turn without significant time penalties. However, since all the buffers are empty at the beginning of the convolution, the algorithm must insure that the buffer levels will naturally and rapidly reach this staircase configuration.

The above is a dynamic problem which cannot be answered merely by analyzing the algorithm utilized. A simulation is needed because confidence in the architecture is desired before drawing any detailed plans of the system. A simulation at the architectural level will efficiently provide the answer to the above buffer-servicing problem.

7.2 Simulator Design

7.2.1 Servicing Algorithm

The discussion of Chapters 5 and 6 reveals that each buffer can store at most 64 units (8 bits wide for input FIFO's and 16 bits wide for the output buffer). Two flags indicate the Almost Full (AF) and Almost Empty (AE) conditions. The AF and AE conditions occur when 56 or more units are present, and when only 8 units or less remains in the buffer respectively. A simple servicing algorithm consists of requesting a DMA transfer to fill the input FIFO when it reaches the AE condition, or to empty the output buffer in the AF state. These transfers will stop only when the input FIFO achieves the AF condition or when the output buffer attains the AE state, irrespective of other DMA requests. When DMA transfers are requested simultaneously, the requesting FIFO with highest priority will be serviced first. The output buffer has higher priority than the input FIFO's to insure proper removal of outputs.

7.2.2 Definition of Operations

The architectural simulator studies only the effects of the presence of buffers between the Convolver and the DMA subsystems. All other supporting subsystems are assumed to add no supplementary delays. Furthermore, the DMA is assumed never to lose the bus. The DMA reading and writing times with the buffers are postulated as short as possible: only six

of its clock cycles (Tdma) are taken. In addition, no time penalty is assumed to be incurred by the DMA when it switches servicing from one FIFO to another. This is true for the 82258 DMA only when DMA requests are detected during servicing and the bus is not relinguished. Inputs and outputs have to be serially shifted into the Convolver processor which means that in an optimal design, at least sixteen of its clock cycles (Tcvlr) are necessary to read from the input buffers, calculate one convolution, and write the previous result to the output FIFO's. In fact, "wait states" may be required by other hardware limitations such as dynamic memory refresh which are not considered in the simulation.

In this simplified system, the Convolver can only operate in two modes. It is either actively computing sums and products or choking due to lack of inputs or to lack of space in the output buffer. When active, each operation takes one unit from the input buffers and adds one unit to the output buffer in sixteen Tcvlr's. Choking is equivalent to an indefinite wait operation. The DMA has five different modes of operation. It can service the output buffer by removing one unit in six Tdma's, or service one of the three input buffers by adding two units in six Tdma's, or the DMA can be idle when there is no DMA request. The DMA idle mode resembles the Convolver "choking" mode since it is equivalent to an indefinite wait operation .

7.2.3 Selection Between Prediction and Emulation Approach

The global state of the convolution board is the combination of the operating mode of its two subsystems. A total of nine different states can be attained during the convolution process. Only the Choking-Idle state is impossible during convolution. In fact, it corresponds to the termination of the convolution.

Using the system simplifications outlined in the previous section, the configuration of

the system buffers at the next change of state can be predicted from their configuration at the beginning of the current state. However, this analysis of the change of state times is very complex since it depends on the contents of each FIFO. The contents of the FIFO's are not simply predictable because of the coupling between the Convolver and the DMA subsystems. Most important of all, this prediction approach requires a great deal of time for the programmer to correctly analyze the behavior of the system and develop the desired program. For this reason another method, the emulation approach, was selected.

In the emulation approach, each subsystem operation is simulated as occurring in a discrete time increment step. The general approach reproducing the stepwise operations makes use of a priority queue data structure[1] in selecting the next future discrete events.[56] Each event is the completion of a single operation on the buffers by either of the two subsystems. Each event will modify the system's current state according to a set of rules, possibly generating additional future events. This approach increases simulation time but greatly simplifies program definition since it simply emulates the actual operation of the system. It also permits an easy modification of the simulator to match changes in the actual system.

7.2.4 Selection Rule for Next Event

The priority queue is a data structure which is used to select the next event among many competing future events. In the case of our convolution board, only the next events from the Convolver and the DMA are contesting for execution. Thus the priority queue reduces to a simple comparison between completion times of the next event of each of the two subsystems. The total elapsed time for each subsystem is kept in variables TSdma and TScvlr respectively. The time interval for the completion of the corresponding next event operations are placed in variables OPdma and OPcvlr. If (TSdma+OPdma) is smaller, greater or equal to (TScvlr+OPcvlr), the event associated with the DMA, with the Convolver

or with both subsystems will occur respectively. The corresponding TS is incremented by the amount of time taken by the operation and the subsystem decides which operation will be performed next. It updates the corresponding OP variable before this decision process is repeated.

The DMA and the Convolver select the following operation to be executed by applying the servicing algorithm or the simple choking rule respectively. This decision is based on the configuration of the FIFO's at completion of the last event. Special care must be taken for the execution of the Idle or Choking operations. Their indefinite wait nature is translated into an "infinite" value for their execution time (OP). The cooperation of the other subsystem is employed to terminate the event which would otherwise take an "infinite" time. Only a change in the configuration of the buffers (eg. full or empty, almost full etc...) can remove the cause for an Idle or Choking event. As soon as one subsystems removes this cause, it will also terminate the indefinite wait operation of the other system. The other subsystem will then be able to select a new operation and participate again in the decision process.

7.2.5 Simulator Implementation

Although some languages have recently been developed for architectural simulations,[46],[67] they lack the dynamic graphical display of selected system information. Furthermore, following the algorithm described above, the simulator is so easily programmed that it was implemented with a simple PASCAL program, written and run under MSDOS on an IBM PC.

The program keeps track of minimum, maximum, and average choking and idle times, in addition to the configuration of the FIFO's and the total elapsed time of each subsystems. The total statistics of the system and the content of the FIFO's can also be shown

dynamically on the computer screen. The content of the buffers were displayed using bar graphs.

7.3 Simulation Results

7.3.1 Total Processing Time

The performed simulations confirmed that the buffers and the servicing algorithm do provide an efficient interface between the Convolver and the DMA. The cyclical staircase configuration is obtained very rapidly by the system and is maintained throughout the rest of the computations.

The maximum operating frequency of the convolution cells is about 20 MHz as discussed in Chapter 8, but the ADMA maximum operating frequency is 16 MHz. This limits the minimum processing time of the DMA to about 246 ms for a 512x512 image. For a range of frequencies between 12 and 20 MHz, the minimum processing time of the Convolver varies from 350 down to 210 ms. The "ideal" processing time of the convolution card becomes the maximum of these two minima while the "actual" time shows the execution time obtained by simulating the interactions due the Convolver frequency, FIFO sizes, and servicing algorithm. Table 8.1 presents these minimum and total processing times for different simulated Convolver frequencies.

Frequency (MHz)	12	14	16	18	20
Time DMA (msec)	245.76	245.76	245.76	245.76	245.76
Time CVLR (msec)	349.53	299.59	262.14	233.02	209.72
Ideal Time (msec)	349.53	299.59	262.14	245.76	245.76
Actual Time (msec)	360.71	317.77	284.87	259.57	245.83
Percentage Increase	3.20	6.07	8.67	5.62	0.03

Table 8.1. Convolution Times Analysis

The percentage increase shows the relative degradation on the actual board design when compared to the "idealized" system whose throughput is limited by the slowest subsystem only. As expected, the simulated times exceed the idealized times but by less than 10 percent, showing the effectiveness of the proposed design. A four-pass 9x9 convolution can still be done in a little more than a second.

7.3.2 Choking and Idle Times

Table 8.2 shows the minimum, average, and maximum DMA idle and Convolver choking time intervals for different Convolver frequencies. The total amount of time spent choking or being idle as well as the number of times this operation was performed are also listed.

Frequency (MHz)	12	14	16	18	20
DMA Idle Times					
Minimum (usec)	27.63	17.73	9.75	3.29	2.03
Average (usec)	28.49	18.12	10.13	3.68	12.90
Maximum (usec)	52.00	42.86	35.63	29.92	22.48
Total (msec)	114.92	71.98	39.08	13.78	0.04
Number Occurrences	4034	3973	3856	3746	3
Convolver Choking Times					
Minimum (usec)	2.46	4.55	5.88	7.07	9.53
Average (usec)	2.76	4.56	5.88	7.08	9.91
Maximum (usec)	21.38	21.38	21.38	21.76	22.48
Total (msec)	11.4	18.13	22.69	26.52	36.09
Number Occurrences	4034	3973	3856	3746	3642

Table 8.2. Idle and Choking Times Analysis

As the Convolver frequency increases, the amount of time spent choking increases whereas the amount of time being idle decreases. For slower Convolver clock frequencies, the increase in total processing time can be compensated by interleaving CPU operations with the DMA memory transfers. Each time the DMA is idle, the workstation's CPU could regain the

MULTIBUS and perform useful work. The simulations show that the portion of the bus bandwidth available to the CPU during convolution processing varies from over 30 percent to almost nil. However, these values would be meaningless if the idle time intervals are too short for any productive operation of the CPU. Fortunately, simulation results show that each idle time interval is at least of the order of several micro-seconds, permitting tens of CPU operations to be performed. Furthermore, as the Convolver frequency decreases, the size and number of idle time intervals increases. Without the simulations, this interesting trade-off between convolution speed and CPU operation would have remained hidden.

8. TESTING RESULTS OF THE VLSI PROCESSOR

8.1 Introduction

The field of ICs has undergone exponential growth particularly in the number of gates placed on one piece of silicon. The increased complexity of circuits can result in large scale test problems especially in the VLSI domain. It is thus very important to detect a fault as early as possible; test effectiveness and high fault coverages at IC level can save a lot of money at the board level and at the system level[37].

Testing an IC means comparing that IC with something that is known to be correct. Comparing the behaviour of the IC with the function specification is referred to as **FUNCTIONAL** testing while comparing the realized structure of the IC (gates and connections) with the structure as defined in the logic description is known as **STRUCTURAL** testing.

The main emphasis of VLSI testing is to orient designers of VLSI chips and boards to think about testing problems in parallel with the design process. With the growing complexity of VLSI systems, their testing is becoming even more complex and almost impossible in many cases. Thus, in VLSI testing one will have to consider structured design-for-testability[94],[91],[62] as a necessary requirement for designing complex systems. The emerging concept of **built-in self-test (BIST)**[92],[82] is very important.

The objective of testing is to ensure that the device will perform all its design functions as defined in specification. The scope of such tests includes characterization testing, production testing, burn-in testing and incoming inspection. Characterization testing determines and tabulates the device characteristics under various ranges of operating parameters such as voltage, frequency or loading. The main objective in production line testing is to regulate

the manufacturing process to ensure that the device meets its specifications and functions correctly based on a suitably quick testing strategy. Here statistical sampling methods are often necessary. In burn-in testing, the idea is to operate the devices for a selected period. After this period, the devices are tested for proper functioning and any failed units are discarded. This strategy is based on the expectation that any defective units will be destroyed during the burn in period. The goal of incoming inspection is to ensure that the combined operation of a complete system functions properly and according to specification.

Characterization testing incorporates the following aspects:

(A). **Electrical Characteristics:** Here the current consumption, output drive capability, input leakage current, range of operating voltage levels, etc. are tested. These are *steady state* characteristics of the device.

(B). **Switching Characteristics:** These document the propagation delay time, setup time, hold time, rise/fall time, access/refresh time, minimum clock width and speed. AC parametric testing is concerned with *timing relationships* as the transistors in the device change their states.

The functional tests of the designed convolution processor were performed using the HP and ASIX 2 test stations. Test vectors were randomly generated for 1024 different possible input combinations and the corresponding expected output vectors were computed. For the HP test station, the HP computer was used to access the input and expected output vectors from the respective disc files and to transmit these values to both the data generator and the logic analyzer which compares these values to those obtained from the test device. Any errors are decoded and stored in a file for further study.

The characterization tests were performed on the following modules shown in Figure 4.2 of the processor architecture block diagram:

(i). C_{in} to C_{out}, i.e. testing of an 8-bit shift register used for coefficient loading and storing.

(ii). X_{in} to X_{out}, i.e. testing of a 32-bit shift register used to load and store image pixel intensities.

(iii). Y_{in} to Y_{out}, i.e. testing of an adder circuit used to sum the computed product with the in-coming partial sum and a 16-bit shift register used to store and shift the computed partial convolution sum.

(iv). Serial-parallel multiplier, i.e. testing of the multiplication of the in-coming image pixel intensity (X_{in}) and the pre-stored coefficient (C_{in}) for proper computation.

(v). OV_{in} to OV_{out}, i.e. testing of the overflow detection circuit.

Examination of the convolution processor function with changes of supply voltage, operating frequency, clock delay, etc. were carefully analyzed and these results are summarized in the following sections.

8.2 Coefficient Loading (C_{in} to C_{out})

Characterization testing was performed on the coefficient loading and storing circuit which includes an 8-bit shift register. A set of 1024 input and output test vectors were generated randomly and saved on a disc file. Then these vectors were loaded into the data

generator. Then using fast binary transfer method, the HP computer loaded the decoded data to the logic analyzer and verified the results obtained for proper matching. The error map facility of the test station was also used to detect errors.

The characterization tests are performed by verifying that the loading functioned correctly. Measurements were taken by fixing some parameters while varying parameter of interest and vice versa. The switching times were verified during the functional test. The **binary search**[5] method was used in this characterization.

The characterization results for this module are shown graphically in Figure 8.1 and Figure 8.2. The module operates over a frequency range of 30 MHz from 10MHz to 40MHz for supply voltages ranging from 4 to 6 volts with Clock Delays (CD) of up to 25ns. Also note that the maximum frequency of operation increases with the supply voltage within the CMOS3 DLM functional voltage range (3 to 7 Volts).

The HP logic analyzer was set for synchronous sampling using an external clock source. This means that the point at which the analyzer samples data is referenced to an external clock. This point can be varied via the **Clock Delay (CD)** setting which is measured from the rising edge of ϕ_2. Four different CD settings were used in this module namely 0ns, 10ns, 25ns, and 50ns. In asynchronous operation, the data is sampled at a fixed point determined by the internal crystal clock generator.

Characterization of maximum clock delay with supply voltage at a fixed frequency of operation was also performed for this module with the results shown in Figure 8.2. Note that the maximum clock delay decreases with increasing supply voltage and that the chip becomes unfunctional outside the rated supply voltage range of 3 to 7 volts. The maximum clock delay must be less than 95% of the clock period and therefore the chip does not function

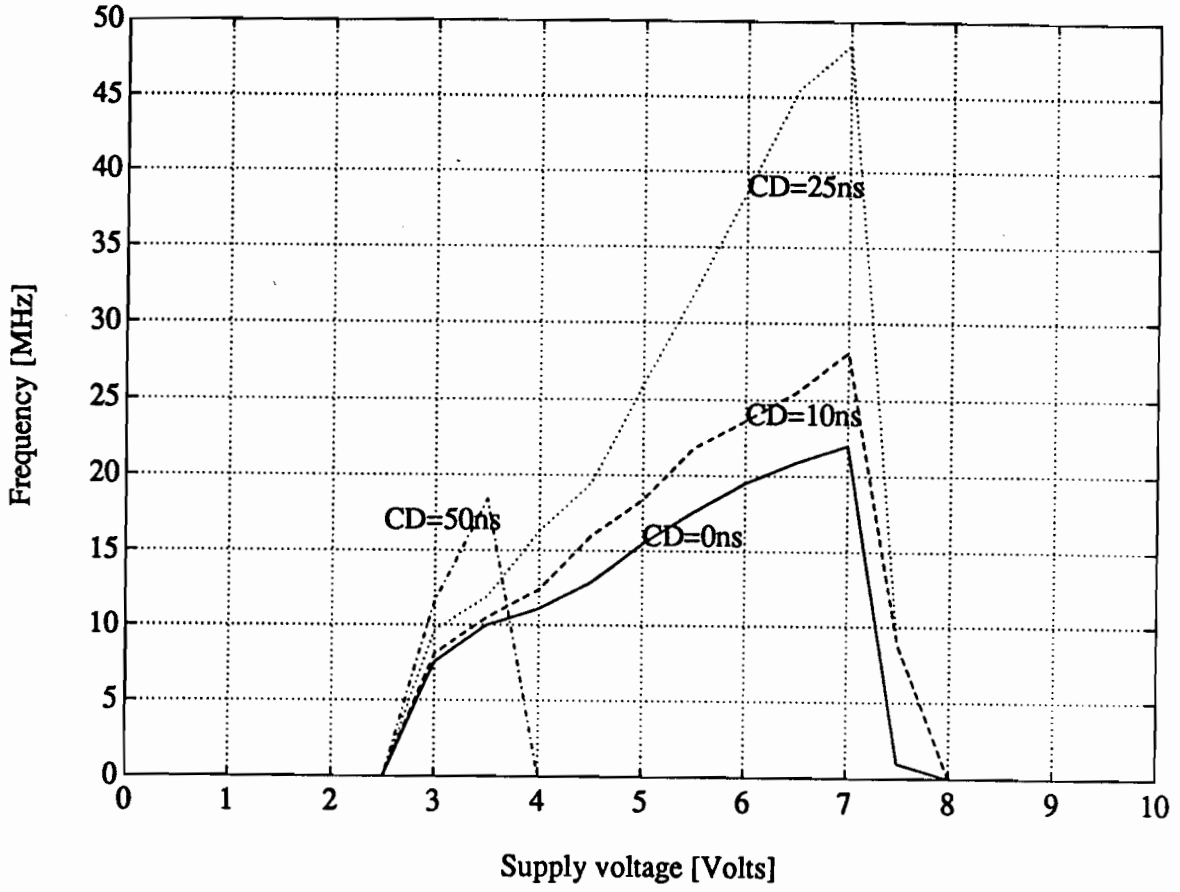


Figure 8.1 Maximum Frequency Vs. Supply Voltage for a Coefficient Loading Circuit.

with CD greater than 50ns (see Figure 8.1).

8.3 Pixel Intensity Loading (X_{in} to X_{out})

The test procedures were repeated to characterize the image pixel intensity loading and storing module containing a 32-bit shift register. Similar results were obtained as summarized in Figure 8.3 and Figure 8.4.

8.4 Partial Sum Module (Y_{in} to Y_{out})

This module contains the adder circuit which computes the partial sum between the in-

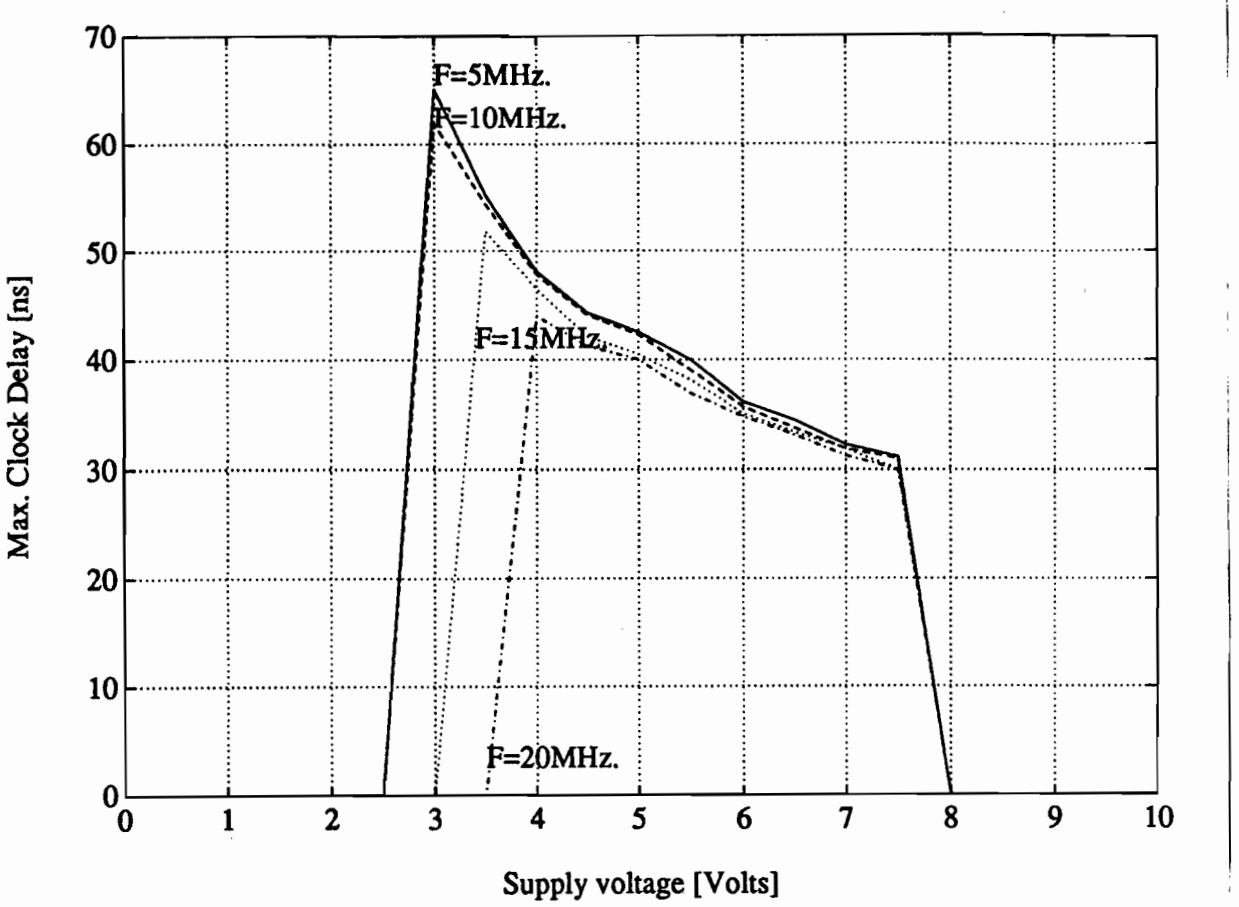


Figure 8.2 Maximum Clock Delay Vs. Supply Voltage for a Coefficient Loading Circuit.

coming sum Y_{in} and the product $C_{in}.X_{in}$ from the serial-parallel multiplier circuit giving the partial sum Y_{out} for the next processor. This module also comprises a 16-bit shift register used to store or shift the computed partial sum. The results of its characterization were similar to that of the image pixel intensity loading circuit shown in Figure 8.3 and 8.4.

8.5 Serial-Parallel Multiplier Module

This circuit contains 8 identical multiplier cells, one for each coefficient bit. The function of each cell is to multiply the in-coming image pixel intensity with the pre-stored coefficient bit. The characterization was found to be unsatisfactory as the expected results were not

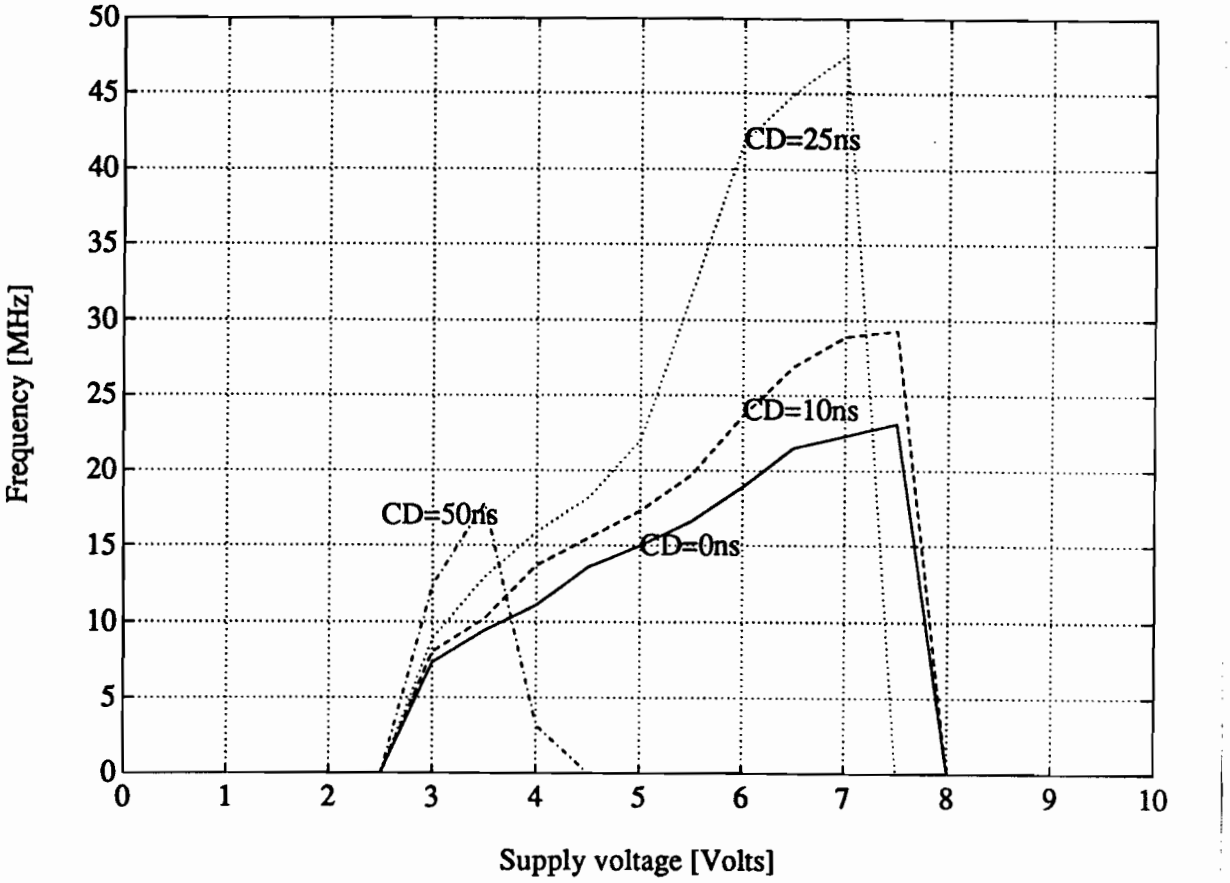


Figure 8.3 Maximum Frequency Vs. Supply Voltage for an Image Pixel Intensity Loading Circuit.

achieved. Identical results were obtained using both the HP and ASIX 2 test stations. Five chips were found to behave the same way suggesting either a design problem or a fabrication problem. It was decided to re-check the design layouts and the corresponding schematics in the SDA environment. The design was originally prepared in the KIRK environment which did not offer the powerful software testing capabilities for design verification currently offered in the SDA environment.

With SDA tools, it was possible to compare all the layouts and schematics for each design module. Schematics were simulated using SILOS, a Logic and Fault Simulator, while layouts were simulated using SPICE. Both of these simulation results were found to compare

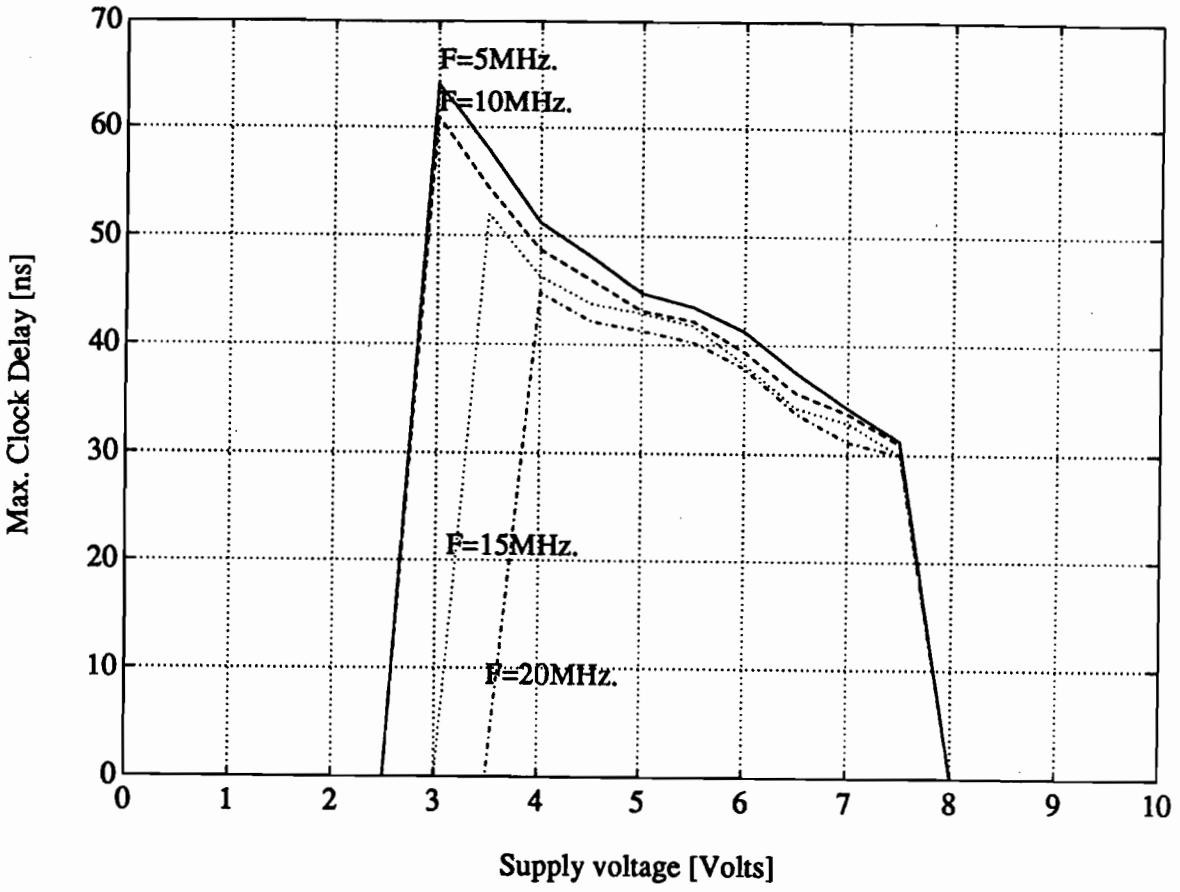


Figure 8.4 Maximum Clock Delay Vs. Supply Voltage for an Image Pixel Intensity Loading Circuit.

succesfully. All cells were found to function as specified in the design specifications. Finally, it was decided to compare the chip schematic and the overall layout including all cells. Here it was detected that the serial-parallel multiplier is not functioning properly because its control signal, **ResetM** was permanently grounded by a fault contact in the layout. Since, the multiplier needs to be reset after every 16 clock cycles by the control signal **ResetM** going "HIGH", this condition was not met and hence the multiplier was not giving the expected results. With this minor correction in the layout the overall schematic was simulated and found to function properly. The chip has now been re-submitted for fabrication.

9. CONCLUSIONS

The designed VLSI convolution system is expected to be the heart of the vision processor to be incorporated in an Intel System 310 running iRMX-286 real-time multitasking operating system. This will allow users to perform convolution of images with 512x512 pixels in about a second for a single 9x9 window size or kernel. For smaller image kernels, the computation time would be proportionally less. This solution is very attractive, especially when compared with the software implementation which takes hours on a Vax Computer for the same kernel.

The design of a convolution system for image processing in a robot vision system is briefly outlined in this thesis. Firstly, the design of a VLSI systolic convolution circuit is presented. The convolution system was implemented using CMOS3 DLM process, i.e. a double well, double metal CMOS process with a 3-micron minimum feature size. The circuit uses multi-processor techniques such as pipelining to divide the computational task among a number of identical processors. Thus, an N-point convolution can be realized using N of the processors operating in pipeline mode. The designed processors may be used to compute any linear, shift-invariant filters and correlation since their equations are very similar to convolution.

To reduce the number of I/O pins required for each chip, bit-serial processing and communication techniques were used throughout the chip design. A total of only 30 pins were used for three systolic processors placed on one 40-pin dual in-line package (DIP) chip. The use of shift-registers to hold the coefficients and their availability as outputs makes it possible to chain them and reduce the board's I/O pin requirements.

The input signals were represented as 8-bit, unsigned numbers (0 to 255), while the

coefficients were represented as 8-bit, signed numbers (-128 to +127). This allows easy interfacing with the existing facilities operating on 8-bit images, and giving a reasonable range of coefficient values. Error detection was provided through the overflow detection circuit, which indicates the occurrence of errors while computing the convolution sums. According to simulation and testing results the convolution processor can operate at a clock frequency of 20 MHz, allowing the processing of 512x512 images for a 9x9 kernel with the designed card in less than a second.

Secondly, the design and performance simulation of a convolution bus card is briefly presented in this thesis. The design of a high performance convolution card was accomplished using architectural simulation as a design tool for digital systems. Due to the programmability of the new Intel 82258 Advanced DMA Controller, pictures of arbitrary sizes can be processed. The basic processing element in this system is the designed VLSI systolic convolution processor. The system is DMA driven allowing the CPU to perform concurrent processing. A prototype board has been constructed interfacing to the MULTIBUS of an Intel System 310 running the Intel iRMX-286 real-time operating system.

Future developments of this research include the design of a new VLSI systolic convolution processor which can handle floating point numbers for more precision. This double precision floating point convolution processor design is expected to greatly facilitate range data image processing studies. Image processing software for frame grabbing of TV Camera images and their display on a RGB color display is also being developed for the RMX environment. Based on such powerful image processing hardware system, it is possible to envisage sophisticated sensor based intelligent or expert robot vision systems in the near future.

REFERENCES

[1] Aho, A.V., Hopcroft, J. and Ullman, J.D., Data Structures and Algorithms, Don Mills: Addison-Wesley Publishing Company, 1983.

[2] Ahuja, N. and Swamy, S., "Multiprocessor Pyramids for Bottom-Up Image Analysis", Proceedings of the International Conference on Pattern Recognition and Image Processing", pp. 380-385, 1982.

[3] Alvertos, N., Brzakovic, D. and Gonzalez, R.C., "Camera Geometries for Image Matching in 3-D Machine Vision", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11, #9**, pp. 897-915, September, 1989.

[4] Armstrong, J.R., "Chip-Level Modeling with HDLs", *IEEE Design and Test of Computers*, pp. 8-18, February 1988.

[5] Baglez, F., "On Testability Analysis of Combinational Networks", 1984 International Symposium on Circuits and Systems, Montreal, Canada, May 7-10, 1984.

[6] Batcher, K.E., "Design of a Massively Parallel Processor", *IEEE Transactions on Computers*, **C-29**, pp. 836-840, 1980.

[7] Besl, P.J. and Jain, R.C., "Invariant Surface Characteristics for 3D Object Recognition in Range Images", *Computer Vision, Graphics, and Image Processing*, **33**, pp. 33-80, January 1986.

[8] Blahut, R.E. "Fast Algorithms for Digital Signal Processing", Addison-Wesley, Reading, MA, 1985.

[9] Boudreault, Y. and Malowany, A., "A VLSI Convolver for a Robot Vision System", Proceedings of the Canadian Conference on Very Large Scale Integration, 1986, Montreal, Quebec, pp. 265-270.

[10] Boudreault, Y., "Design of a VLSI Convolver for a Robot Vision System", a thesis submitted to the Faculty of Graduate Studies and Research, McGill University, Montreal, Canada, 1986.

[11] Bromley, K., Symanski, J.J. and Speiser, J.M., "Systolic Array Processor Developments". In VLSI Systems and Computations by H.T.Kung and R.F. Sproul, 1981, pp. 273-284.

[12] Butter, J.T. and Kerkhoff, H.G., "Multiple-Valued CCD Circuits", *IEEE Computer Magazine*, **21, #4,** pp. 58-70, April, 1988.

[13] Capello, P.R. and Steiglitz, K., "Digital Signal Processing Applications of Systolic Algorithms". In VLSI Systems and Computations, by H.T.Kung and R.F. Sproul, 1981, pp. 245-254.

[14] Capello, P., Davidson, G., Gersho, A., Koc, C. and Somayazulu, V., "A Systolic Vector Quantization Processor for Real-time Speed Coding", Proc. IEEE Int. Conf. ASSP, Tokyo, Japan, April 1986.

[15] Capson, D.W. and Eng, S.K., "A Tiered-Color Illumination Approach for Machine

Inspection of Solder Joints", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **10, #3**, pp. 387-393, May, 1988.

[16] Carayannis, G., Freedman, P. and Malowany, A., "An Integrated programming environment for a generic robotic workcell", Proceedings of the Symposium on Manufacturing Application Languages, June, 1988, Winnipeg, Manitoba, pp. 71-85.

[17] Chakrabarti, C. and Ja'Ja', J., "Optimal Systolic designs for computing DHT and DCT", in VLSI Signal Processing, III, R. Brodersen, and H. Moscovitz, Eds. New York: IEEE Press, pp. 411-422, 1988.

[18] Chang, J.H., Ibarra, O.H., Pong, T.C. and John, S.M., "Two-Dimensional Convolution on a Pyramid Computer", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **10, #4**, pp. 590-594, July, 1988.

[19] Cho, N.I. and Lee, S.U., "DCT Algorithms for VLSI Parallel Implementations", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **387, #1**, pp. 121-128, January, 1990.

[20] Collet, C., Cote, J.F., Haule, D.D. and Malowany, A.S., "Architectural Simulation as a Design Tool for Digital Systems", Proceedings of the 1989 Summer Computer Simulation Conference, Austin, Texas, pp. 151-156, July 24-27, 1989.

[21] Cook, G.E., "Robotic Arc Welding: Research in Sensory Feedback Control", *IEEE Transactions on Industrial Engineering*, **IE-30, #3**, pp. 252, August 1983.

[22] Cookey, M., Trussel, H.S. and Won, I.J., "Seismic Deconvolution by Multiple Meth-

ods", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **38, #1**, pp. 156-160, January, 1990.

[23] Corby, N.R. "Machine Vision for Robotics", *IEEE Transactions on Industrial Engineering*, **IE-30, #3**, pp. 282-291, August 1983.

[24] Cote, C.J., Collet, C., Haule, D.D. and Malowany, A.S. "A High Performance Convolution Processor", SPIE Proceedings of the Visual Communications and Image Processing III, Cambridge, Massachusetts, Volume 1001, pp. 469 - 475, November 9-11, 1988.

[25] Cote, J.F., Larochelle, F. and Malowany, A.S., "Architecture Simulation of VLSI Design to Validate Algorithms", Proceedings of the 1989 Summer Computer Simulation Conference, Austin, Texas, pp. 115-118, July 24-27, 1989.

[26] Cowan, C.K. and Kovesi, P.D., "Automatic Sensor Placement from Vision Task Requirements", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **10, #3**, pp. 407-416, May, 1988.

[27] De Floriani, L., "Feature Extraction from Boundary Models of Three-Dimensional Objects", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11, #8**, pp. 785-799, August, 1989.

[28] Deller, J.R., Jr., "A 'Systolic Array' Formulation of the Optimal Bounding Ellipsoid Algorithm", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37, #9**, pp. 14325-1436, September, 1989.

[29] Dillman, E.G., "Vision System for Quality Control of Label Application", Proceedings of SPIE, Vol. 336, pp. 168, 1984.

[30] Fountain, J.J., Mathews, K.N. and Duff, M.J.B., "The CLIP7A Image Processor", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **10, #3**, pp. 310-320, May, 1988.

[31] Fountain, T.J. and Geotcherian, V., "CLIP4 Parallel Processing System", Proceedings of IEEE, Vol. 127, pp. 219-224, 1980.

[32] Fu, K.S., "Robotics and Automation", *IEEE Computer*, **15, #12**, pp. 34-40, December 1982.

[33] Fu, K.S., "Pattern Recognition for Automatic Visual Inspection", *IEEE Computer*, **15, #12**, pp. 34, December 1982.

[34] Gader, P.D., "Bidiagonal Factorization of Fourier Matrices and Systolic Algorithms for Computing Discrete Fourier Transforms", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37, #8**, pp. 1280-1284, August, 1989.

[35] Gauthier, D., Levine, M.D., Malowany, A., Begnoche, N. and Lefebvre, G., "Measuring the alignment accuracy of surface mount assembly circuit Board Masks", Vision Interface 88, June, 1988, Edmonton, pp. 1-6.

[36] Gennery, D.B., "A Stereo Vision System for an Autonomous Vehicle", Proceedings from The International Joint Conference on Artificial Intelligence, pp. 576-582, 1977.

[37] Goel, P., "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits", *IEEE Transactions on Computers*, **c-33, #3**, pp. 215-222, March 1981.

[38] Goncalves, R.C. and H.J. DeMan, "NORA: A Racefree Dynamic CMOS Technique for Pipelined Logic Structures", *IEEE Journal on Solid State Circuits*, **SC-18, #3**, pp. 261-267, June 1983.

[39] Gu, J. and Smith, K.F., "A Structured Approach for VLSI Circuit Design", *IEEE Computer Magazine*, **22, #11**, pp. 9-23, November, 1989.

[40] Guide to the Integrated Circuit Implementation Services of the Canadian Microelectronics Corporation, Carruthers Hall, Queen's University, Kingston, Canada, GICIS Version 3:0, January 1987.

[41] Hall, E.L., Tio, J.B.K., McPherson, C.A. and Sadjadi, F.A., "Measuring Curved Surfaces for Robot Vision", *IEEE Computer*, **15, #12**, pp. 42-54, December 1982.

[42] Hansen, C. and Henderson, T.C., "CAGD-Based Computer Vision", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11, #11**, pp. 1181-1194, November, 1989.

[43] Harmon, L.D., "Automated Touch Sensing", *International Journal of Robotics Research*, **1, #2**, pp. 3-32, 1982.

[44] Haule, D.D. and Malowany, A.S., "High-speed 2-D Hardware Convolution Architecture Based on VLSI Systolic Arrays", IEEE Proceedings of the 1989 Pacific Rim

Conference on Communications, Computers and Signal Processing, Victoria, BC, pp. 52-55, June 1-2, 1989.

[45] Hwang, J.N., Vlontzos, J.A. and Kung, S.Y., "A Systolic Neural Network Architecture for Hidden Markov Models", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37, #12**, pp. 1967-1980, December, 1989.

[46] Iacobovici, S. and CC. Ng, "VLSI and System Performance Modeling", *IEEE Micro*, **7, #4**, August 1987, pp. 59-72.

[47] Intel Corporation, Intel 82258 ADMA, User's Guide, 3065 Bowers Avenue, Santa Clara, California 95051, 1986.

[48] Krambeck, R.H., Lee, C.M. and Law, H.F.S., "High-Speed Compact Circuits with CMOS", *IEEE Journal on Solid State Circuits*, **SC-17, #3**, pp. 614-619, June, 1982.

[49] Kung, H.T., "Why Systolic Architectures?", *IEEE Computers*, **15, #1**, pp. 37-46, January 1982.

[50] Kung, H.T. and Picard, R.L., "Hardware Pipelines for Multi-Dimensional Convolution and Resampling", *CAPADM-81*, pp. 273-278, 1981.

[51] Kung, H.T. and Song, S.W., "A Systolic 2-D Convolution Chip", *CAPADM-81*, pp. 159-160, 1981.

[52] Kung, H.T., "Why Systolic Architectures", *IEEE Computer*, Jan. 1982, pp. 37-46.

[53] Kung, S.Y. and Gal-Ezer, R.J., "Synchronous Versus Asynchronous Computation in Very Large Scale Integrated (VLSI) Array Processors", Proceedings of SPIE, Real Time Signal Processing V, Vol. 341, pp. 53-65, 1982.

[54] Landeta, D. and Malinowski, C.W., "Two Dimensional Convolver Architecture for Real Time Image Processing", Proceedings of the SPIE Visual Communications and Image Processing, Philadelphia, Penn., pp. 1095-1105, Nov. 5-10, 1989.

[55] Larochelle, F., Cote, J.F. and Malowany, A.S., "A Floating Point Convolution Systolic Cell", Proceedings of Vision Interface, London, Ontario, June 19-23, 1989.

[56] Law, A.M. and Kelton, W.D., Simulation Modeling and Analysis, New York: McGraw-Hill Book Company, 1982.

[57] Lee, S.Y. and Aggarwal, J.K., "Parallel 2-D convolution on a mesh connected array processor", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-9**, pp. 590-594, July, 1987.

[58] Lee, S.Y. and Aggarwal, J.K., "A System Design/Scheduling Strategy for Parallel Image Processing", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12, #2**, pp. 194-204, February, 1990.

[59] Levine, M.D., Vision in Man and Machine, New York: McGraw-Hill Book Company, 1985.

[60] Ling, N., Malowany, M.E. and Malowany, A.S., "An Expert System Scheduler for Circuit-Board Repair Tasks in a Robotic Workcell", 1989 ASME International Com-

puters in Engineering Conference, Anaheim, CA, pp. 523-529, July 30 - August 2, 1989.

[61] LSI Logic Corporation, L64240 Multi-bit Filter Data Sheets, 1551 McCarthy Blvd., Milpitas, California 95035, July 1987.

[62] Maamari, F. and Rajski, J., "A Reconvergent Fanout Analysis for Efficient Exact Fault Simulation of Combinational Circuits", Proceedings of the 18th Fault Tolerant Comp. Symposium, pp. 122-126, June 1988.

[63] Malowany, M.E. and Malowany, A.S., "Color-Edge Detectors for a VLSI Convolver", Proceedings of the SPIE Visual Communications and Image Processing, Philadelphia, Penn., pp. 1116-1126, Nov. 5-10, 1989.

[64] Malowany, M.E. and Malowany, A.S., "A Systolic Cell for Fit-Error Computations in Range-Image Processing", Proceedings of the 1989 ASME International Computers in Engineering Conference, Anaheim, California, pp. 71-79, July 30 - August 2, 1989.

[65] Mansouri, A.R. and Malowany, A., "Using Vision Feedback in Printed Circuit Board Assembly", IEEE Microprocessor Forum Conference, Atlantic City, N.J., March, 1985, pp. 115-122.

[66] Michaud, C., Malowany, A.S. and Levine, M.D., "Electronic Assembly by Robots", Graphics Interface 1985, Montreal, Canada, pp. 391-397, May 27-31, 1985.

[67] Mitchell, C.L. and Flynn, M.J., "A Workbench for Computer Architects", *IEEE Design and Testing of Computers*, February 1988, pp.19-29.

[68] Movich, R.C., "Vision-Controlled Robotic Cell", Proceedings of SPIE, Robot Vision, Vol. 336, pp. 59-66, 1984.

[69] Nakagawa, Y., "Automatic Inspection of Solder Joints on Printed Circuit Boards", Proceedings of SPIE, Vol. 336, pp. 121-127, 1984.

[70] Nelson, R.C. and Aloimonos, J., "Obstacle Avoidance Using Flow Field Divergence", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **112, #10**, pp. 1102-1106, October, 1989.

[71] Ni, L.M. and Jain, A.K., "A VLSI Systolic Architecture for Pattern Clustering", *IEEE Trans. Pattern Analysis Machine Intelligent*, **PAMI-7**, pp. 80-89, Jan. 1985.

[72] Nitzan, D., "Three-Dimensional Vision Structure for Robot Applications", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **10, #3**, pp. 291-310, May, 1988.

[73] Oppenheim, A.V. and Willsky, A.S., "Signals and Systems", Prentice-Hall, Englewood Cliffs, NJ, 1983.

[74] Parent, P. and Zucker, S.W., "Trace Inference, Curvature Consistency and Curve Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11, #8**, pp. 823-840, August, 1989.

[75] Parthasarathy, S., Birk, J. and Dessimoz, J., "Laser Rangefinder for Robot Control and Inspection", Proceedings of SPIE, Vol. 336, pp. 2-11, 1984.

[76] Progress Report, Computer Vision and Robotics Laboratory (CVaRL), Department of Electrical Engineering, McGill University, Montreal, Canada, 1988/89.

[77] Qureshi, Q.A. and Fischer, T.R., "A Hardware Processor for Implementing the Pyramid Vector Quantizer", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37, #7**, pp. 1135-1143, July, 1989.

[78] Raibert, M.H. and Tanner, J.E., "Design and Implementation of a VLSI Tactile Sensing Computer", *International Journal of Robotics Research*, **1, #3**, pp. 3-18, Fall 1982.

[79] Reed, T.R. and Wechsler, H., "Segmentation of Textured Images and Gestalt Organization Using Spatial/Spatial-Frequency Representations", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12, #1**, pp. 1-13, January, 1990.

[80] Reinhold, A.G., "Automatic Inspection of Sheet Metal Parts", Proceedings of SPIE, Vol. 336, pp. 84-90, 1984.

[81] Roberts, R.A. and Mullis, C.T., "Digital Signal Processing", Addison-Wesley Publishing Company, Inc., Reading, MA, 1987.

[82] Robinson, M. and Rajski, J., "An Algorithmic Branch and Bound Method for PLA Test Pattern Generation", Proceedings of the 1988 IEEE International Test Conference, pp. 784-795, April 1988.

[83] Roncella, R. and Saletti, R., "A VLSI Systolic Adder for Digital Filtering of Delta-Modulated Signals", *IEEE Transactions on Acoustics, Speech, and Signal Processing*,

37, #5, pp. 749-755, May, 1989.

[84] Sanderson, A.C. and Perry, G., "Sensor-Based Robotic Assembly Systems: Research and Applications in Electronic Manufacturing", Proceedings of IEEE, Vol. 71, N0: 7, pp. 856-871, July 1983.

[85] SDA Systems Inc., "Schematics, Simulations and Tests", Reference manual, Version 2.0, February 1988.

[86] Shahraray, B. and Anderson, B.J., "Optimal Estimation of Contour Properties by Cross-Validated Regularization", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11, #6**, pp. 600-611, June, 1989.

[87] Siegel, H.J., Siegel, L.J., Kemmerer, F.C., Mueller, P.T., Smalley, H.E. and Smith, S.D. "PASM: A Partitionable SIMD/MIMD System for Image Processing and Pattern Recognition", *IEEE Transactions on Computers*, **C-30**, pp. 934-947, December 1981.

[88] Silverman, H.F., "Programming the WFTA for Two-Dimensional Data", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37, #9**, pp. 1425-1431, September, 1989.

[89] Truong, T.K., Reed, I.R., Hsu, I.S., Shyu, H.C. and Shao, H.M., "A pipelined design of a fact prime factor DFTM a finite field", *IEEE Computer Magazine*, **37**, pp. 266-273, March, 1988.

[90] Uhr, L., "Pyramid Multi-Computer Structures and Augmented Pyramids", in Computing Structures for Image Processing, M.J.B. Duff editor, Academic Press, pp. 95-

112, 1983.

[91] Wagner, K.D., Chin, C.K. and McCluskey, E.J., "Pseudorandom Testing", *IEEE Transactions on Computers*, **C-36, #3**, pp. 332-343, March 1987.

[92] Wang, F., "BIST Using Pseudorandom Test Vectors and Signature Analysis", Proceedings of the IEEE Custom Integrated Circuits Conference, pp. 1611-1618, January 1988.

[93] Weste, N. and Eshraghian, K., "Principles of CMOS VLSI Design, A Systems Perspective", Addison-Wesley, 1985.

[94] Williams, T.W. and Parker, K.P., "Design for Testability - A Survey", Proceedings of the IEEE, vol. 71, N0: 1, pp. 98-112, January 1983.

[95] Yakimovsky, Y. and Cunningham, R., "A System for Extracting Three- Dimensional Measurements from a Stereo Pair of TV Cameras", *Computer Graphics and Image Processing*, **7**, pp. 195-210, 1978.

[96] Yeh, H., "Systolic Implementation on Kalman Filters", *IEEE Trans. on ASSP*, **36, #9**, Sept. 1988, pp. 1514-1517.

[97] Yen, D.W. and Kulkarni, A.V., "The ESL Systolic Processor for Signal and Image Processing", Proceedings of the 1981 IEEE Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management, IEEE Computer Society Press, pp. 265-272, November, 1981.

[98] Yeshurun, Y. and Schwarts, E.L., "Shape Description with a Space- Variant Sensor: Algorithms for Scan-Path, Fusion, and Convergence over Multiple Scans", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11, #11**, pp. 1217-1222, November, 1989.

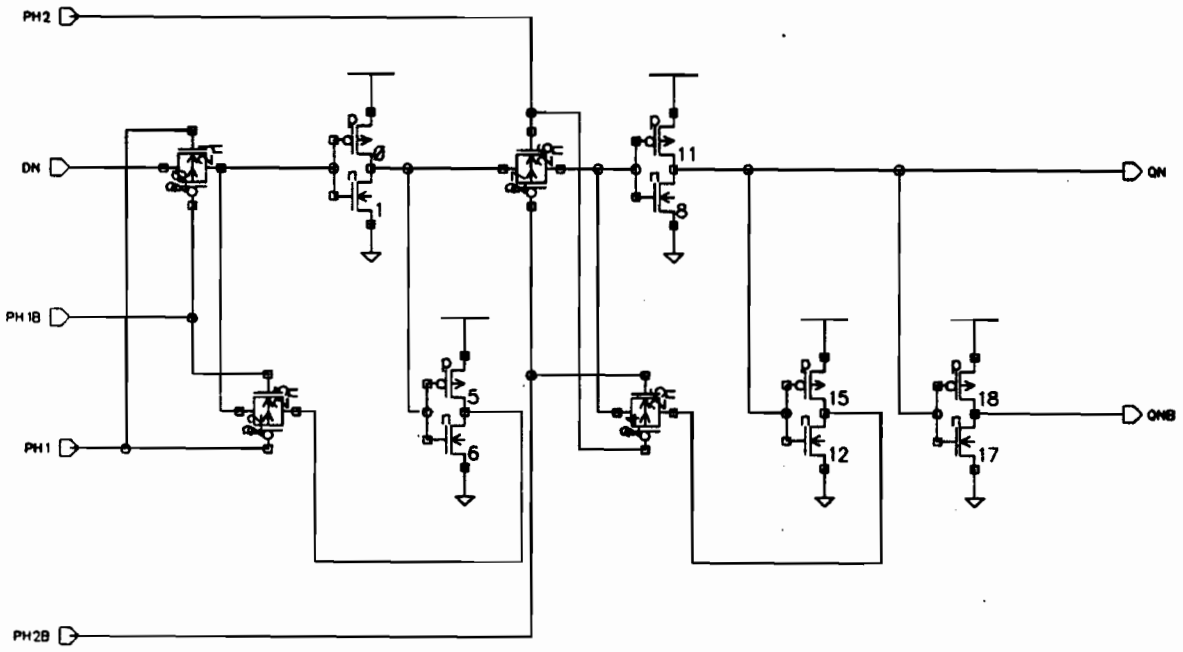
[99] Yokoya, N. and Levine, M.D., "Range Image Segmentation Based on Differential Geometry: A Hybrid Approach", technical report: McRCIM-TR-CIM 87-167, McGill Research Center for Intelligent Machines, McGill University, September 1987.

[100] Zohars, S., "A VLSI Implementation of a Correlator/Digital-Filter Based on Distributed Arithmetic", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37, #1**, pp. 156-161, January, 1989.

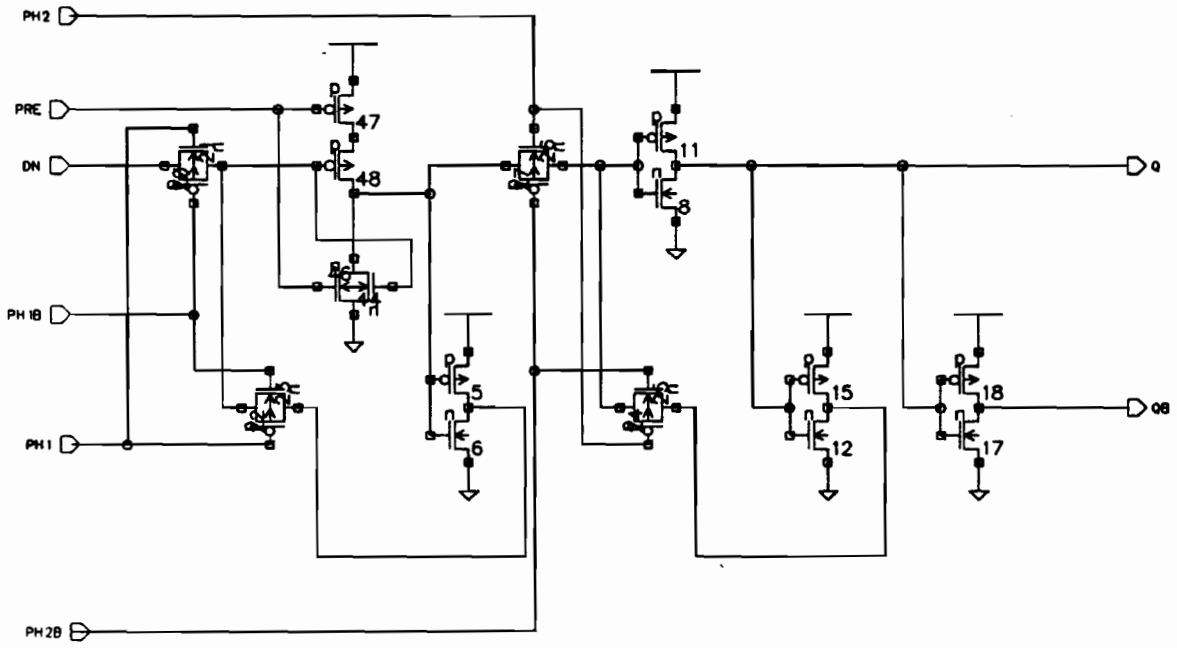
APPENDIX A: SDA SCHEMATICS OF THE PROCESSOR

This page contains 14 Sheets

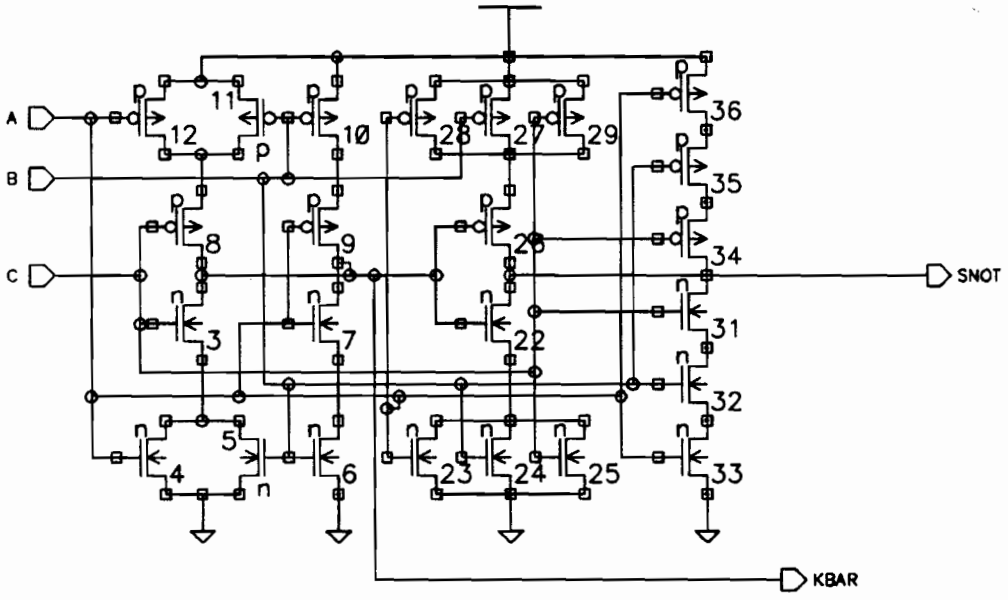
Flip Flop without Pre-set (ffnp)



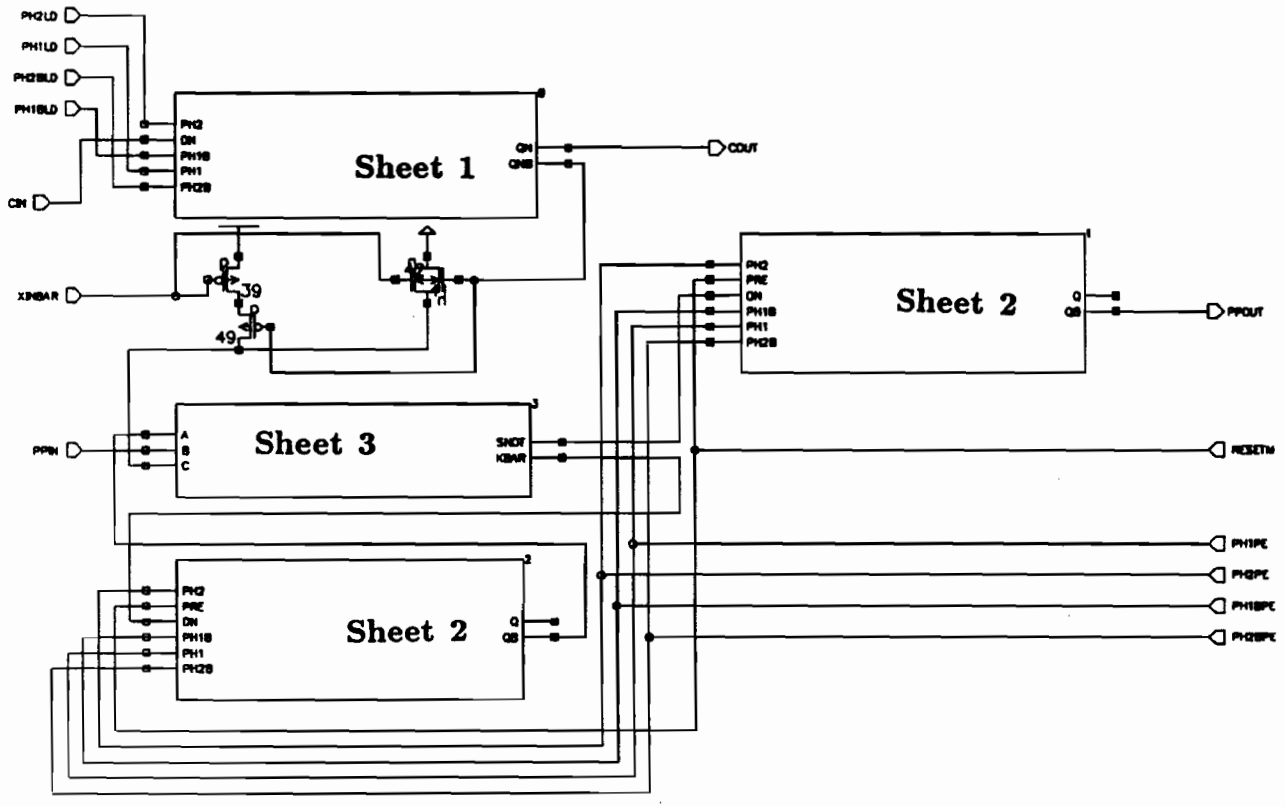
Flip Flop with Pre-set (ff)



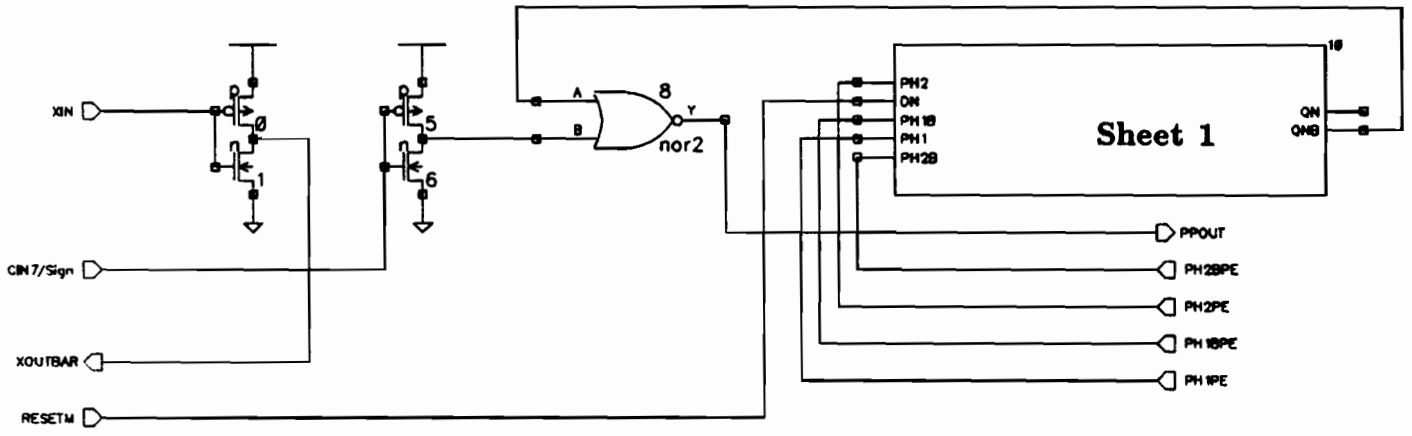
3-input Full-Adder Circuit (fa2)

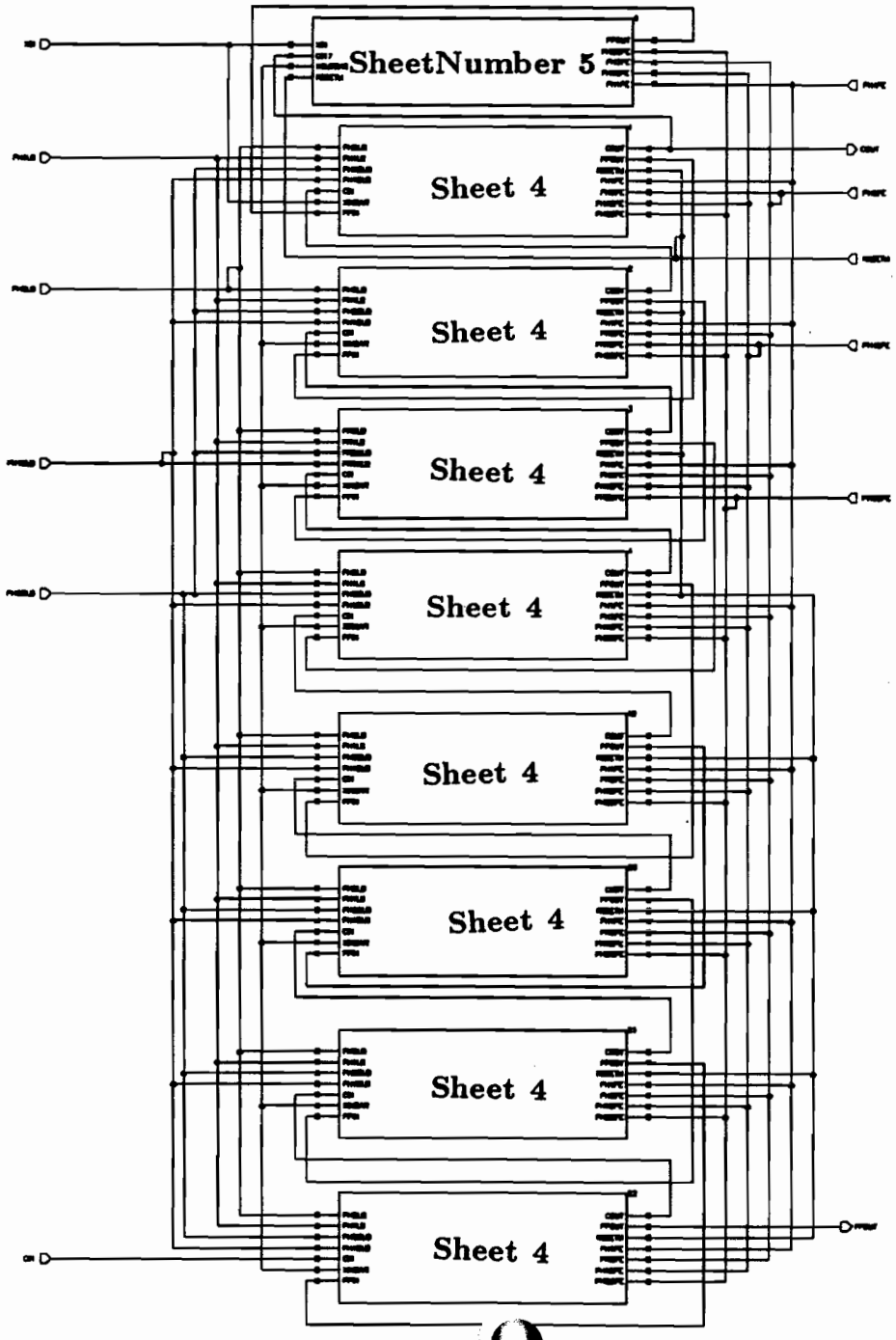


Basic Multiplier Cell (mult)



Control Circuit for the Multiplier (topmult)

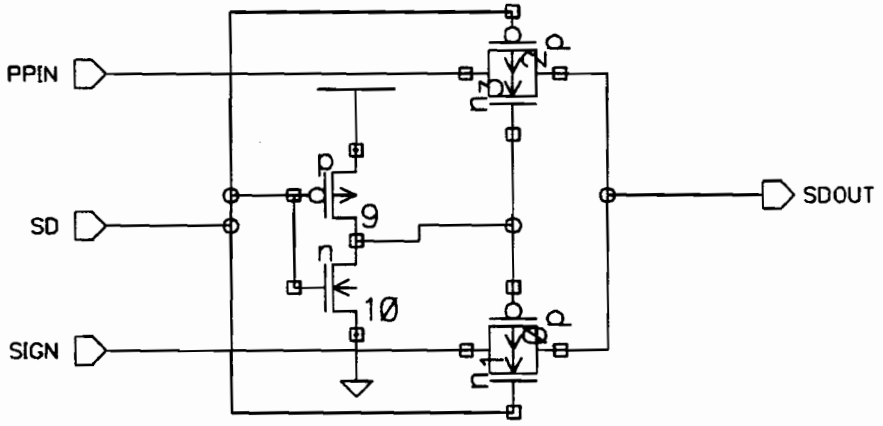




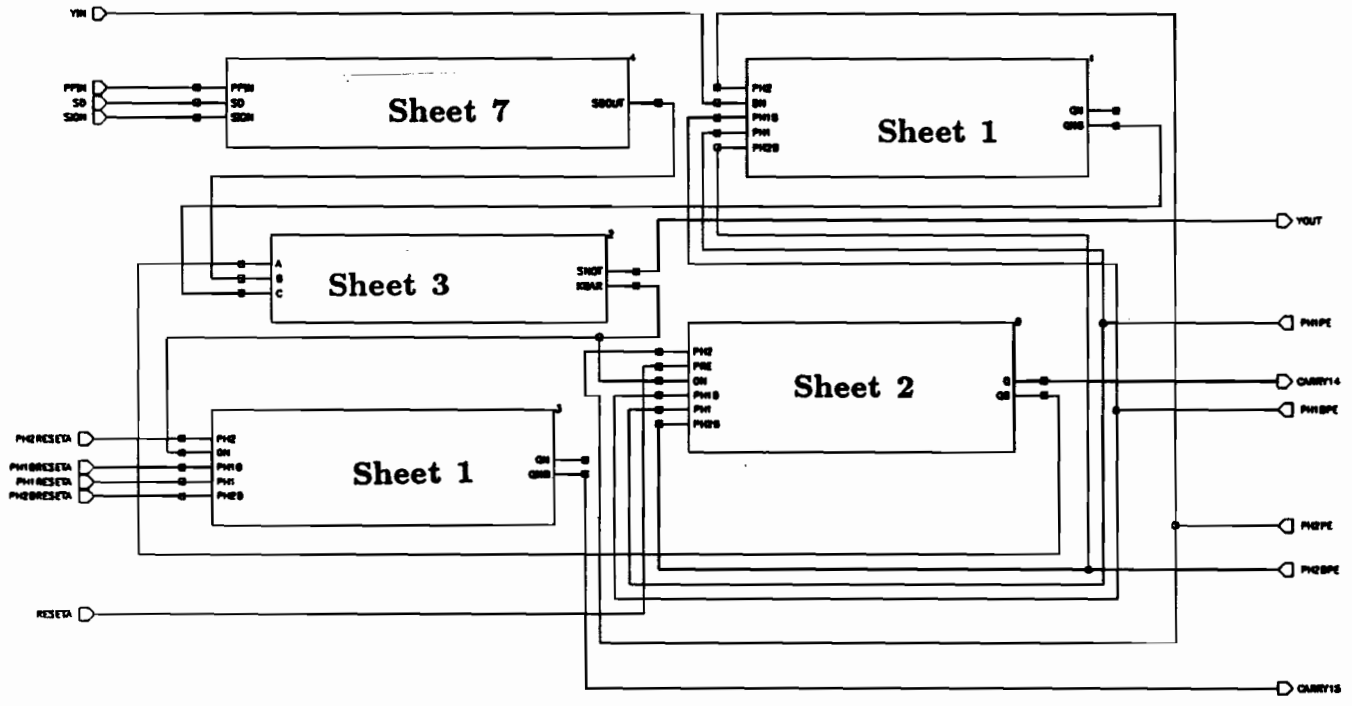
Serial-Parallel Multiplier (8-bits)

SheetNumber 7

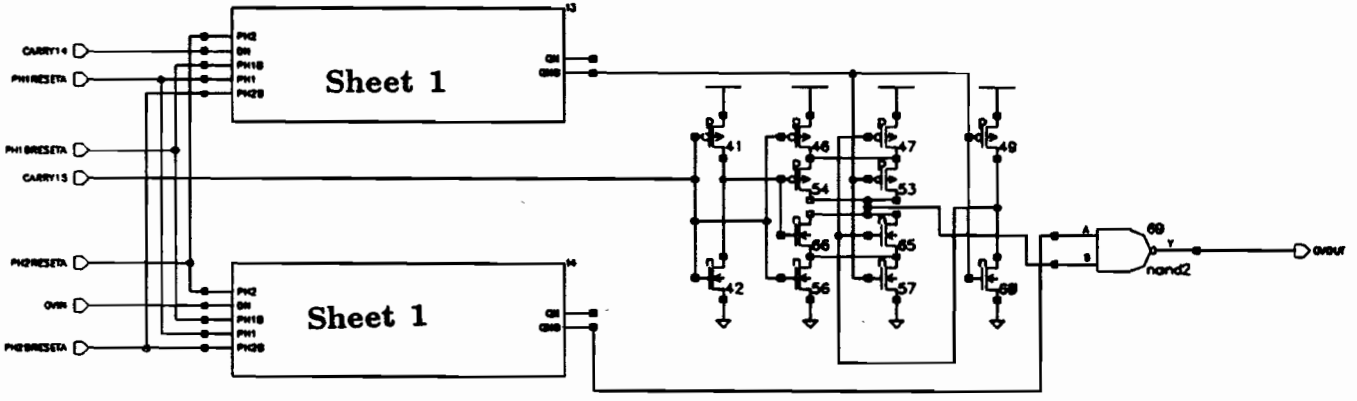
Sum Disable Circuit (sdcircuit)



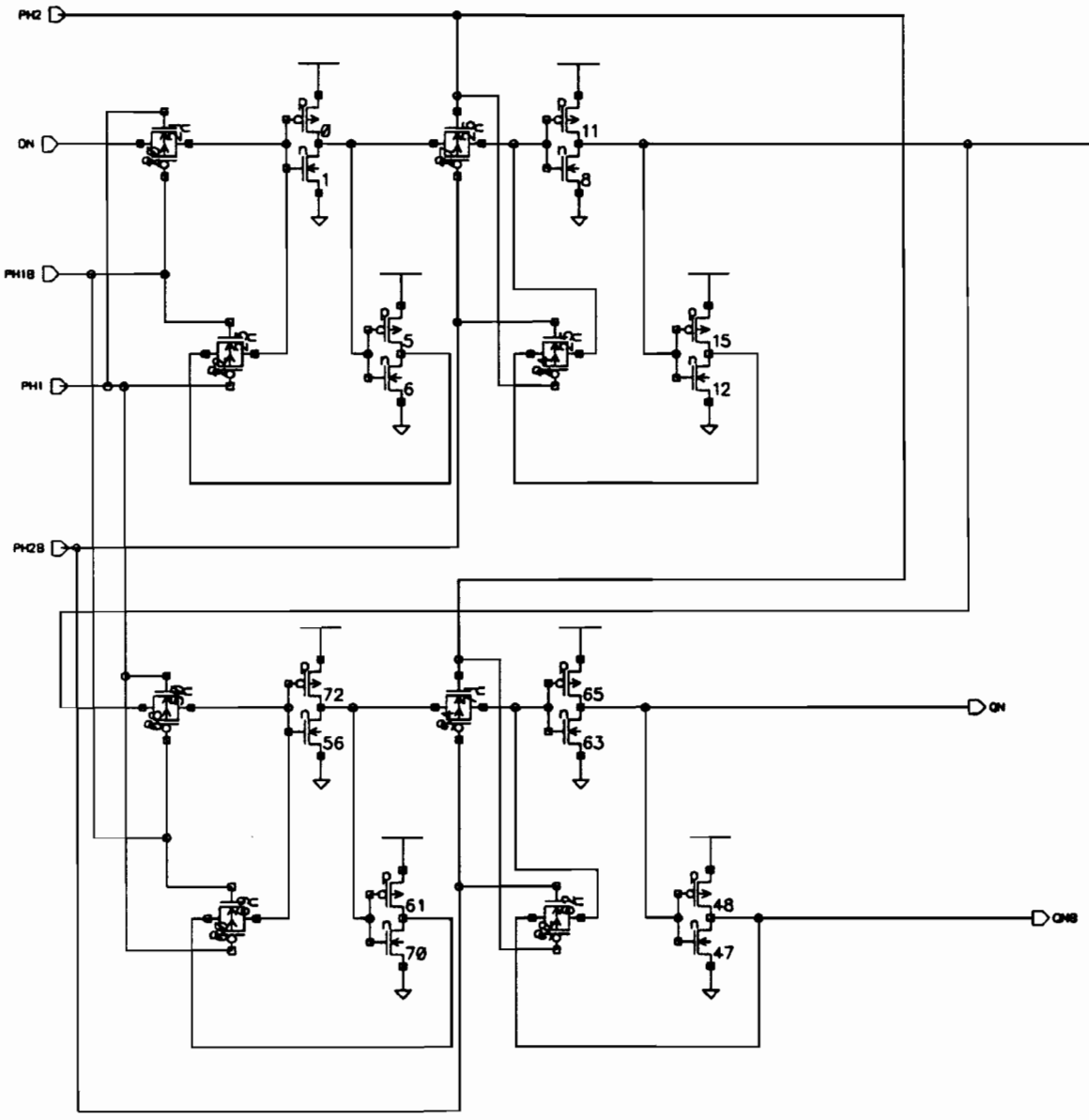
Partial Sum Adder Circuit (add3)



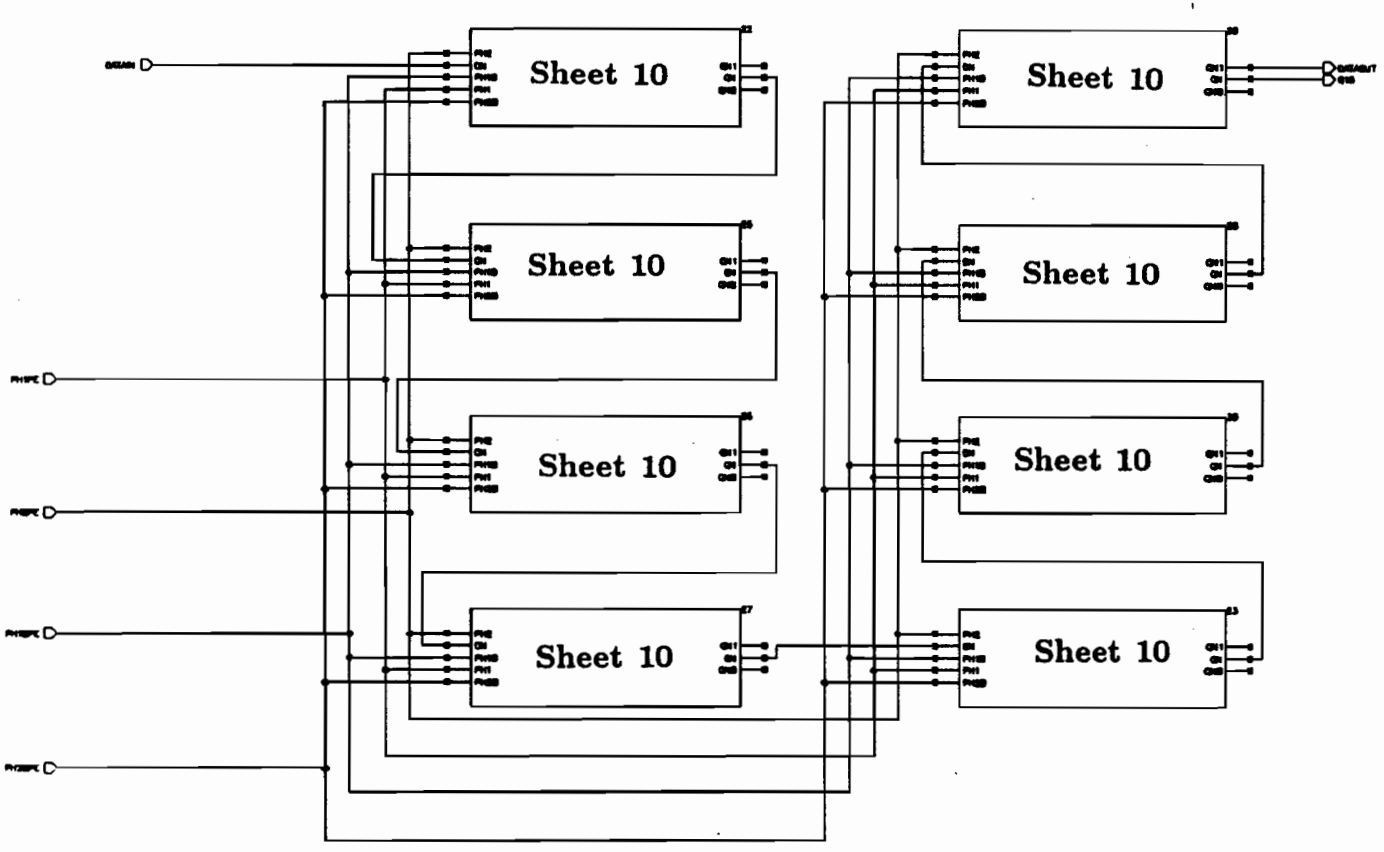
Overflow Detection Circuit (ov2)



2-bit Shift Register (sr) SheetNumber 1Ø

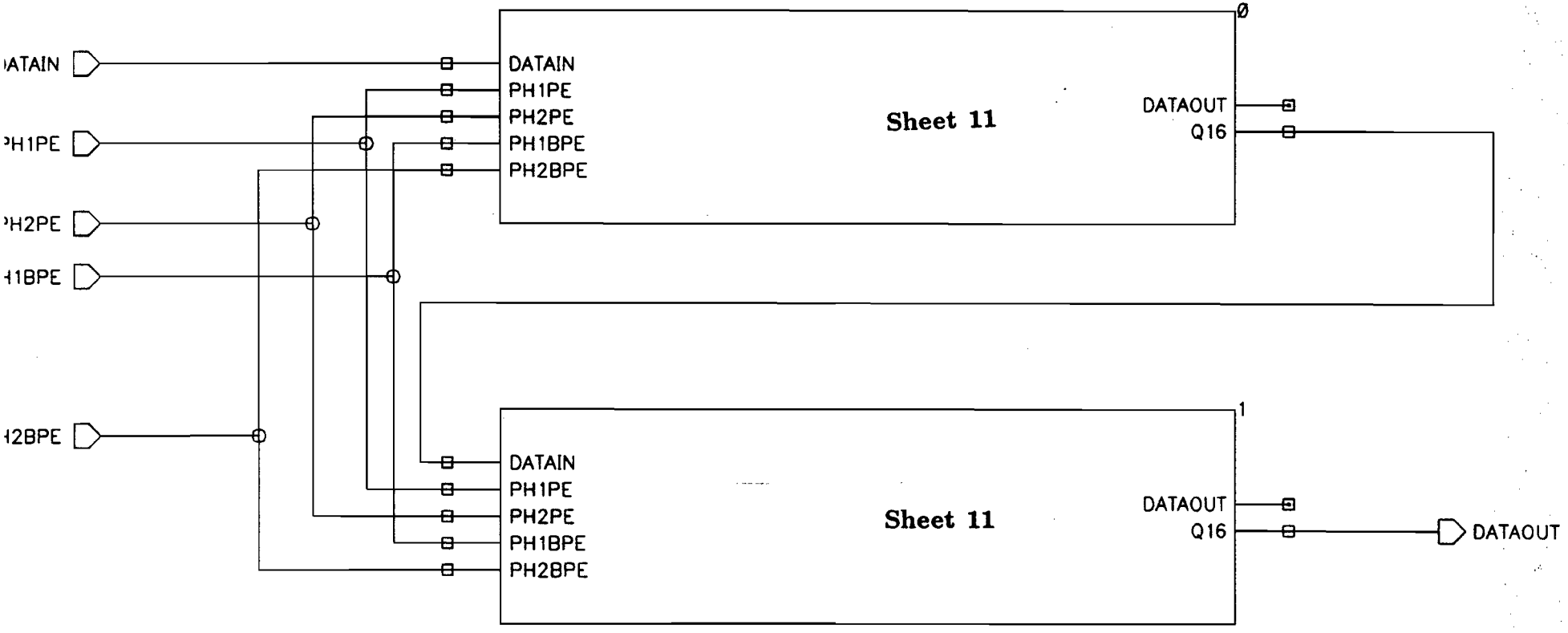


16-bit Shift Register (sr16)



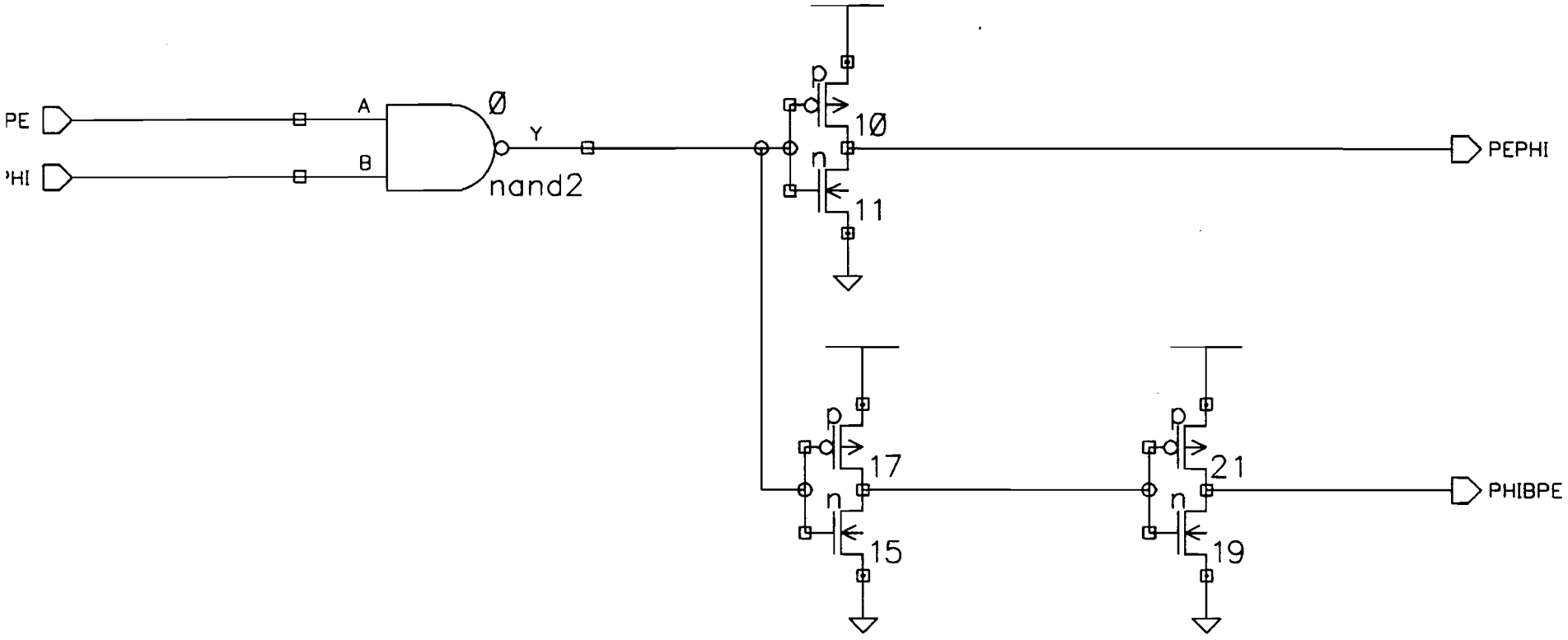
SheetNumber 12

32-bit Shift Register (sr32)



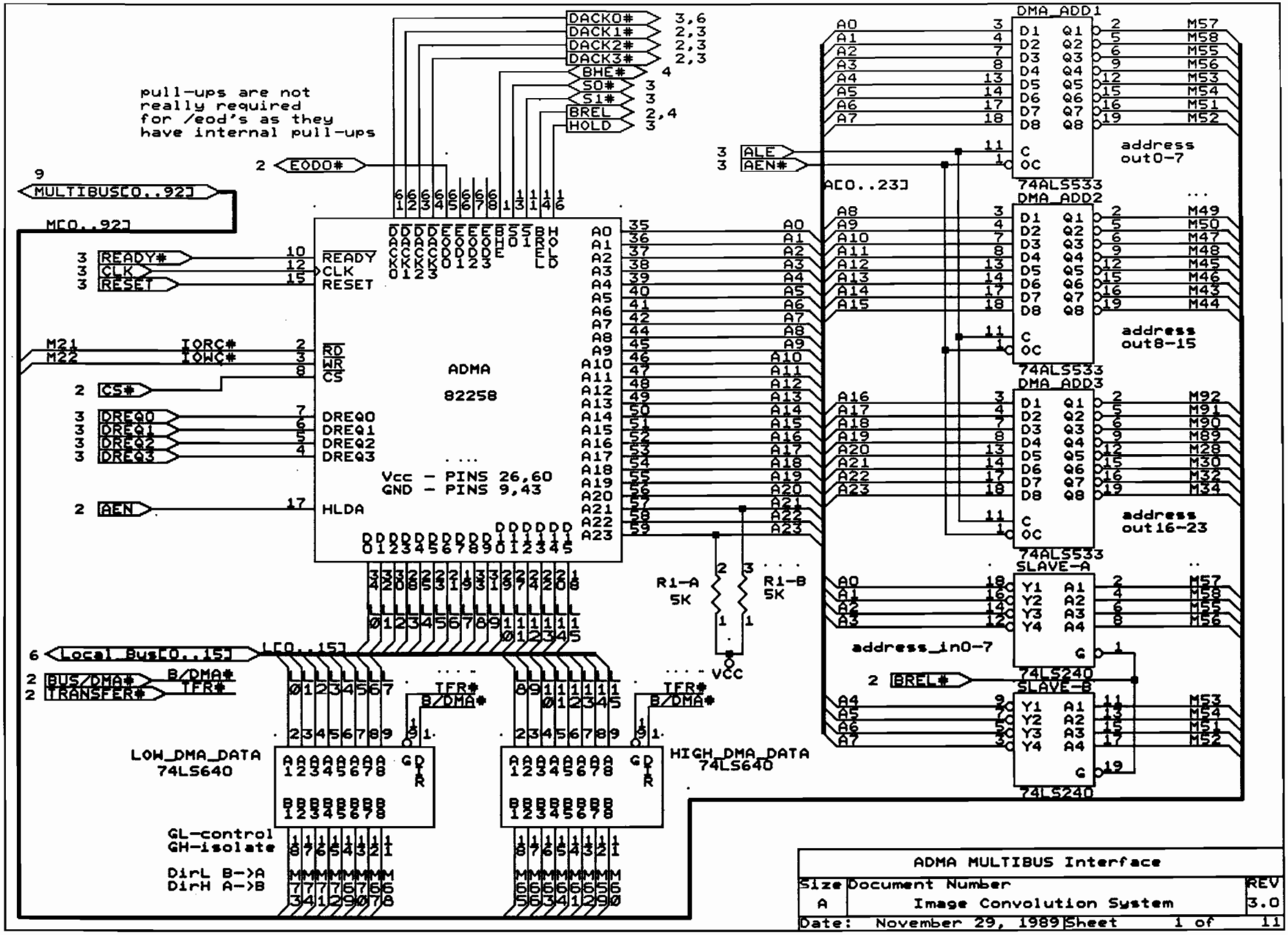
SheetNumber 13

Control Signal Driver Circuit (driv)

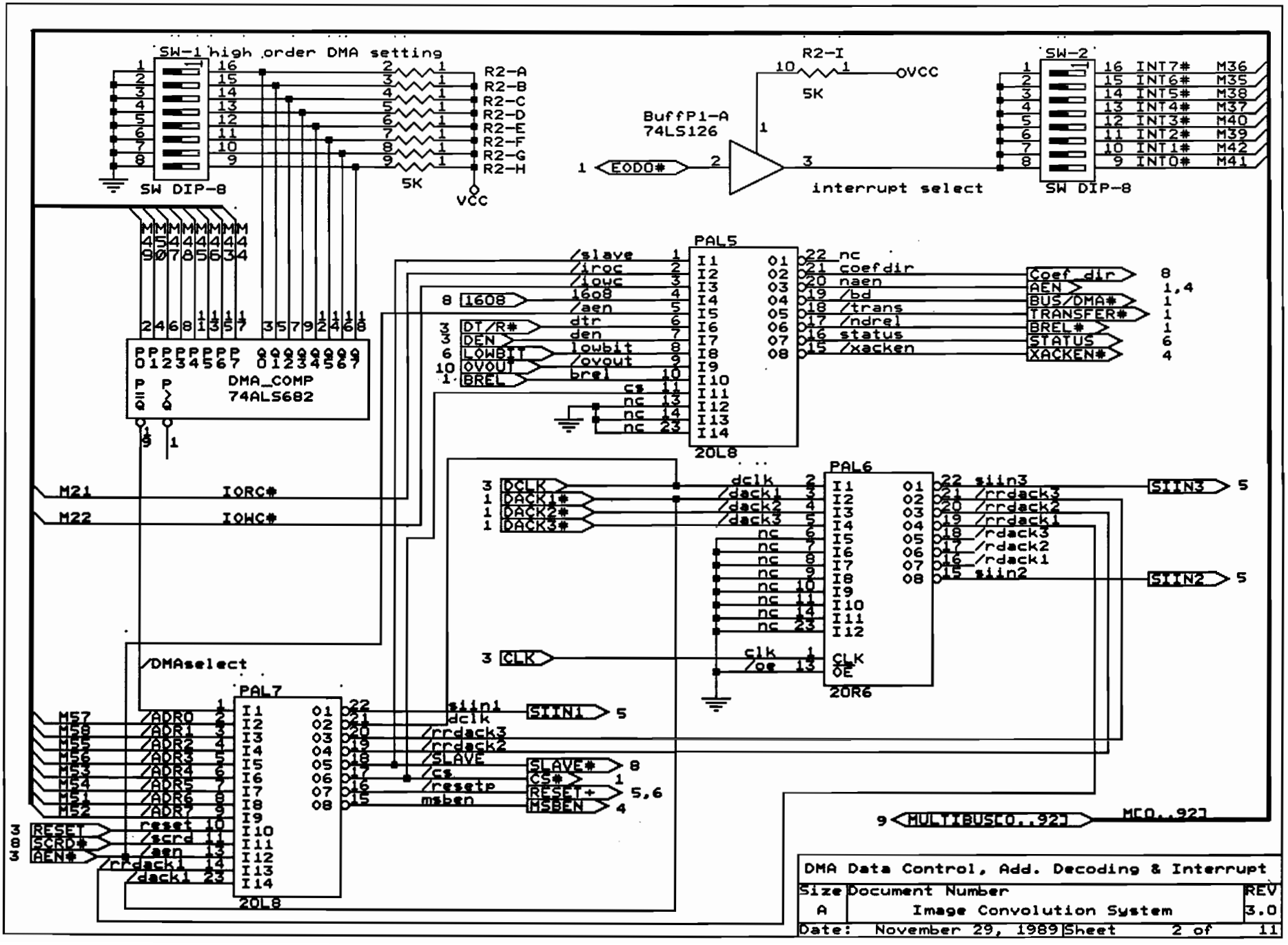


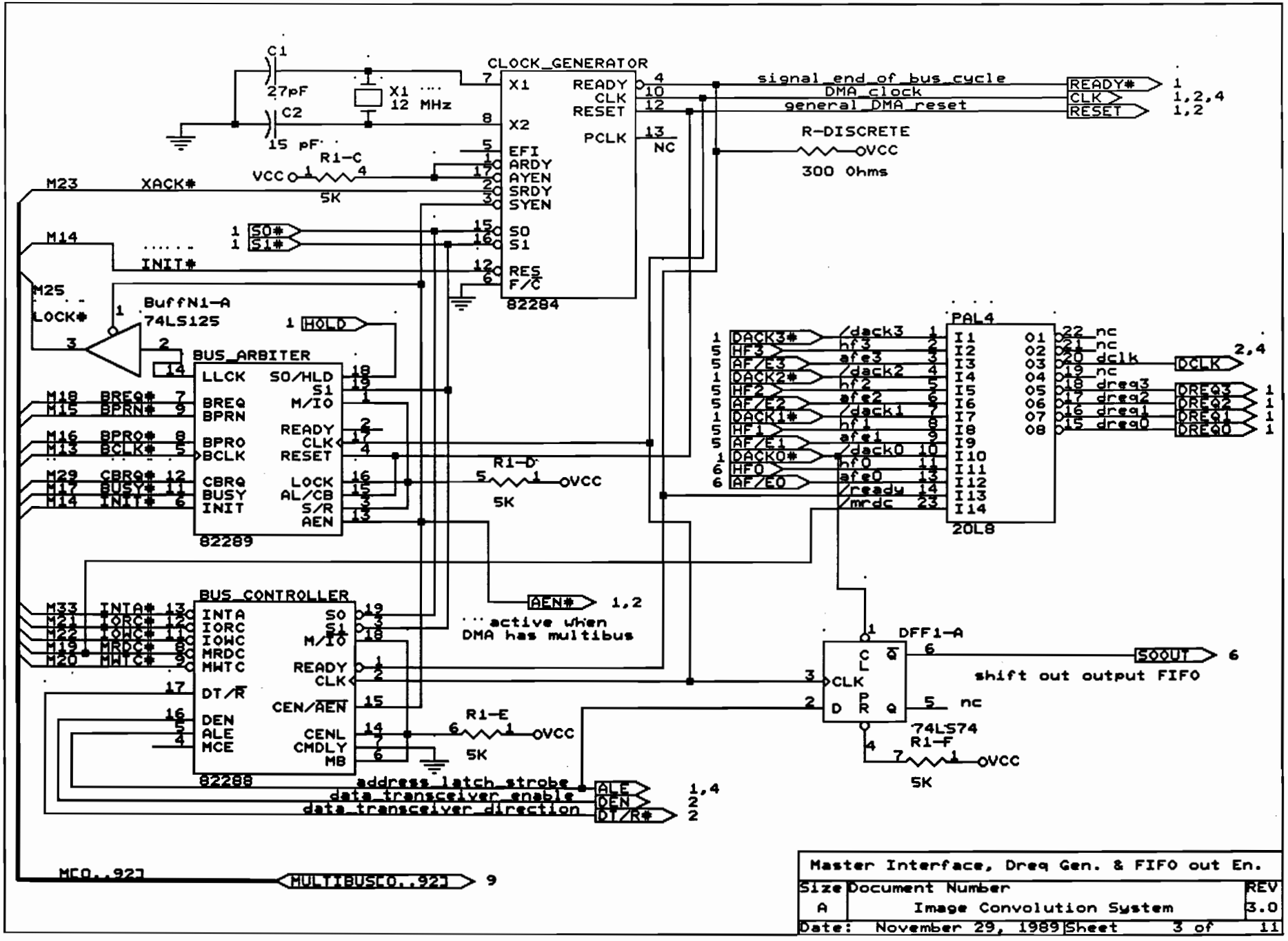
APPENDIX B: ORCARD SCHEMATICS OF THE BOARD

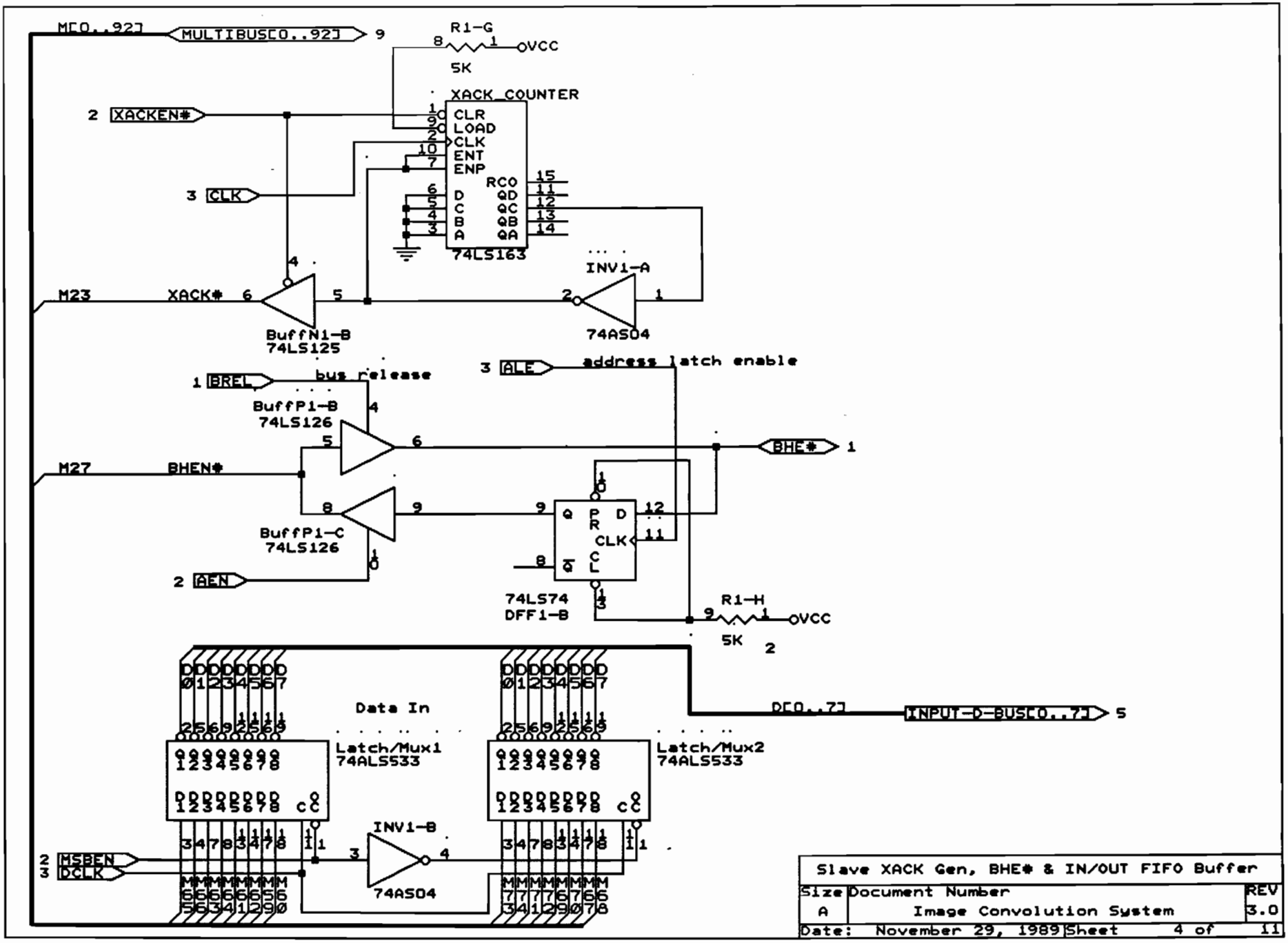
This page contains 10 Sheets

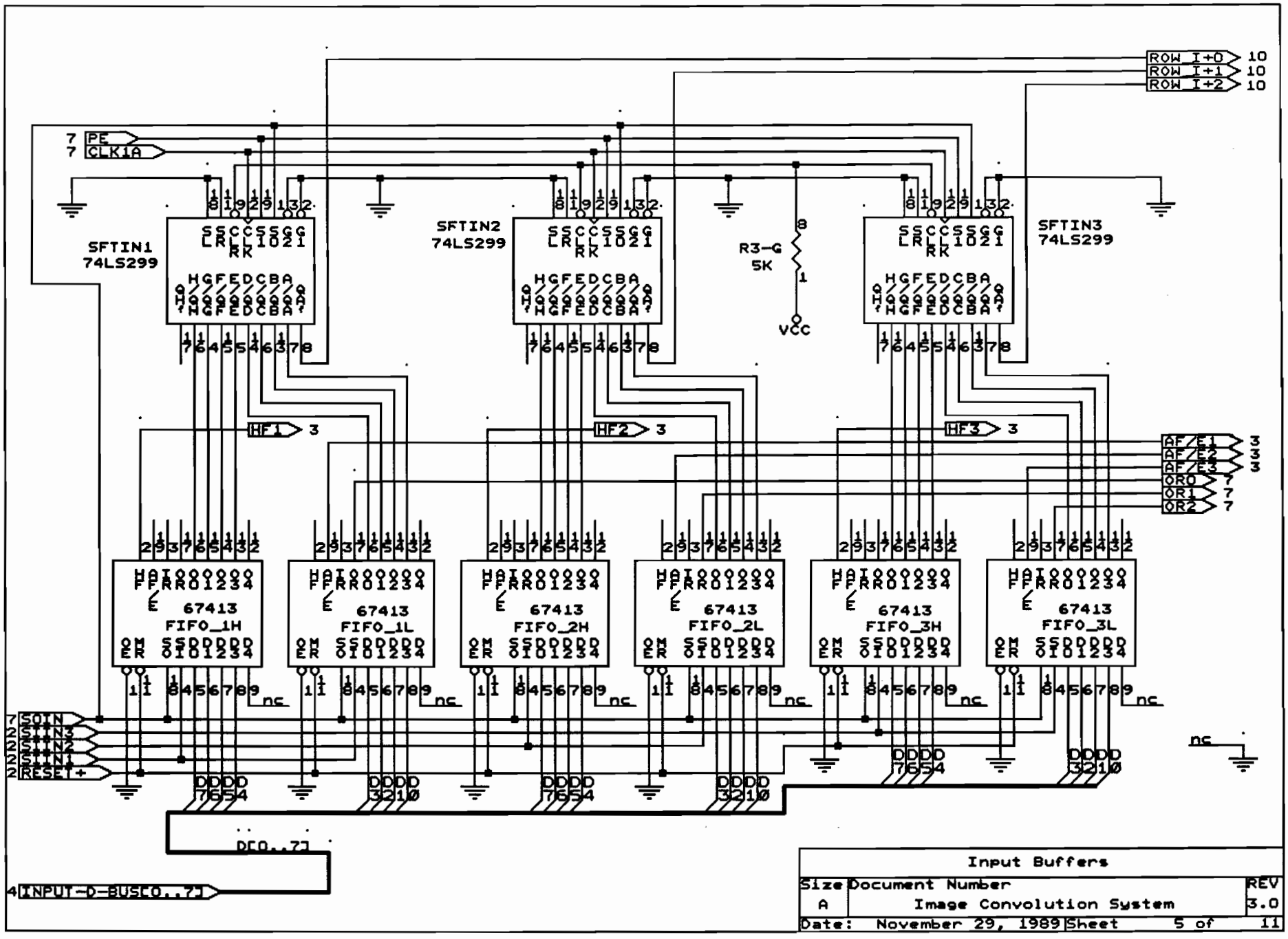


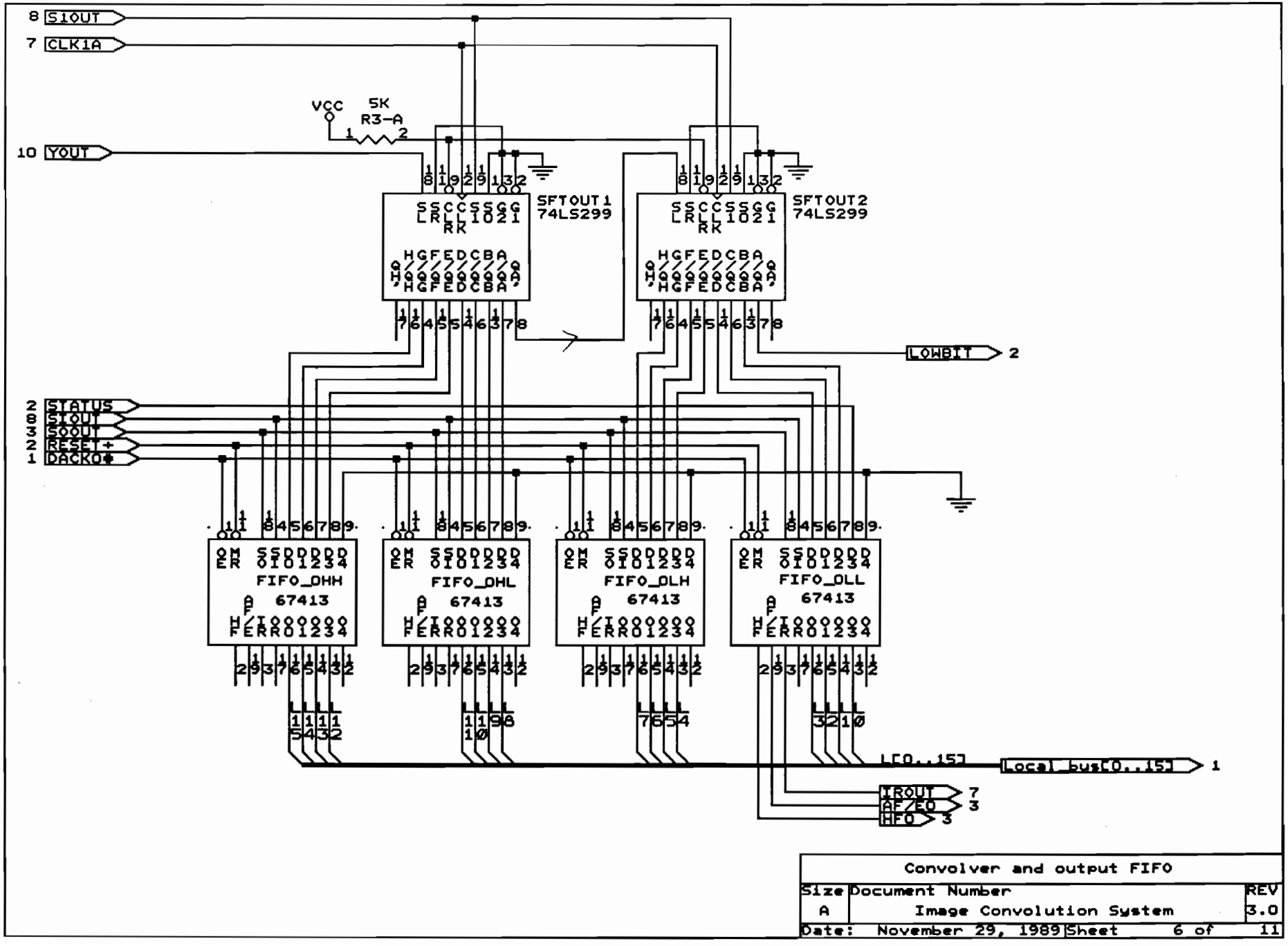
ADMA MULTIBUS Interface		
Size: A	Document Number: Image Convolution System	REV: 3.0
Date: November 29, 1989	Sheet 1 of 11	

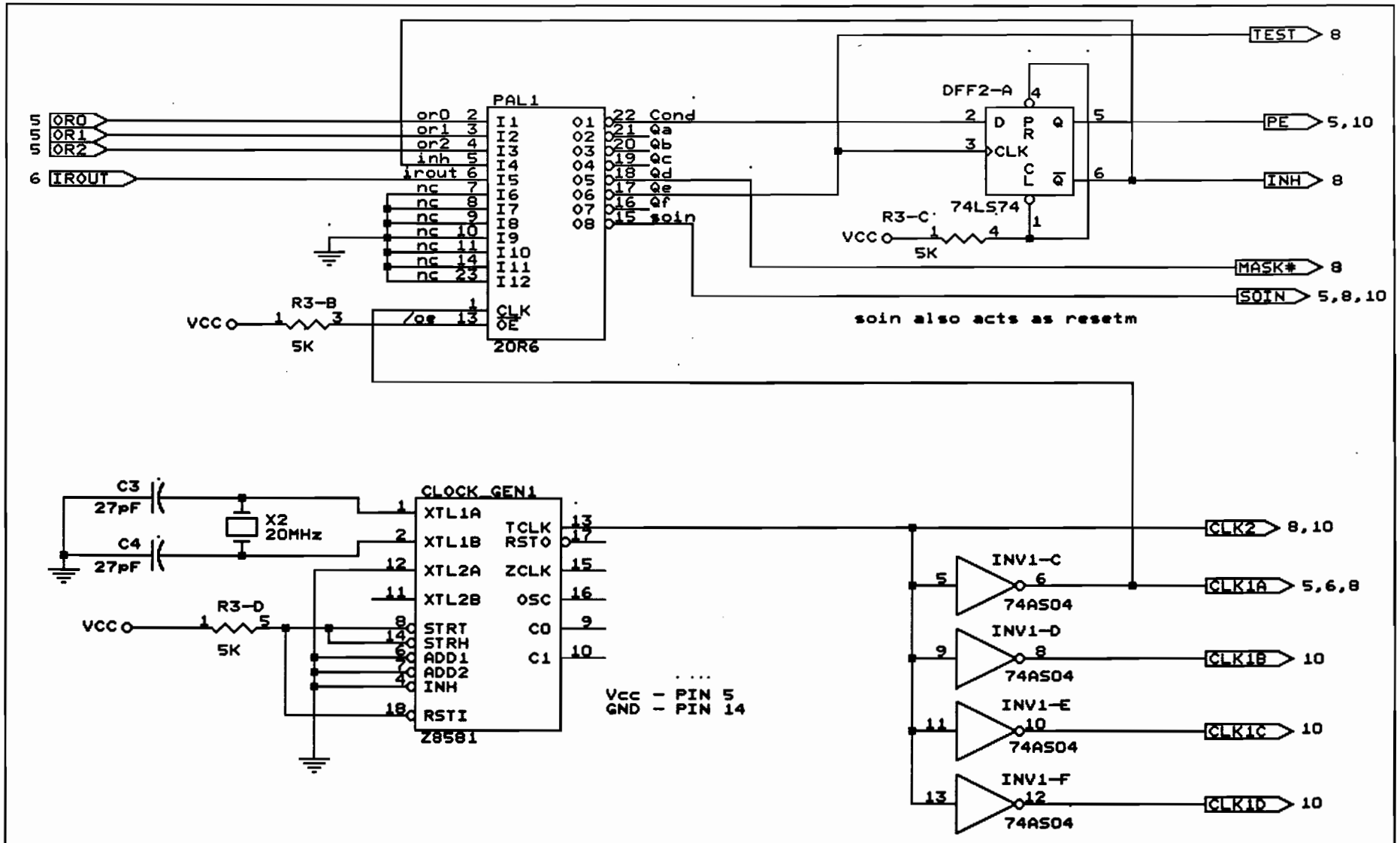

SW-1 high order DMA setting
SW DIP-8
R2-A
R2-B
R2-C
R2-D
R2-E
R2-F
R2-G
R2-H
5K
VCC
R2-I
5K
VCC
BuffP1-A
74LS126
EODO#
interrupt select
SW-2
SW DIP-8
INT7# M36
INT6# M35
INT5# M38
INT4# M37
INT3# M40
INT2# M39
INT1# M42
INT0# M41
DMA_COMP
74ALS682
PAL5
20L8
/slave
/iroc
/iowc
16o8
/aen
dtr
den
lowbit
/ovout
brel
cs
nc
1608
DT/R*
DEN
LOWBIT
OVOUT
BREL
Coef_dir
AEN
BUS/DMA#
TRANSFER#
BREL#
STATUS
XACKEN#
/coefdir
naen
/bd
/trans
/ndrel
status
/xacken
PAL6
20R6
dclk
/dack1
/dack2
/dack3
DCLK
DACK1#
DACK2#
DACK3#
CLK
SIIN3
SIIN2
/rrdack3
/rrdack2
/rrdack1
/rdack3
/rdack2
/rdack1
siin2
siin3
M21
IORC#
M22
IOWC#
DMAselect
PAL7
20L8
ADR0
ADR1
ADR2
ADR3
ADR4
ADR5
ADR6
ADR7
M57
M58
M55
M56
M53
M54
M51
M52
RESET
SCRD#
AEN#
/rrdack1
/dack1
siin1
dclk
/rrdack3
/rrdack2
/SLAVE
/cs
/resetp
msben
SIIN1
SLAVE#
CS#
RESET+
MSBEN
MULTIBUS[0..92]
MD0..92
DMA Data Control, Add. Decoding & Interrupt
Size A
Document Number
Image Convolution System
REV 3.0
Date: November 29, 1989
Sheet 2 of 11


C1
27pF
X1
12 MHz
C2
15 pF
CLOCK_GENERATOR
X1
X2
EFI
ARDY
AYEN
SRDY
SYEN
SO
S1
RES
F/C
READY
CLK
RESET
PCLK
NC
82284
signal_end_of_bus_cycle
DMA_clock
general_DMA_reset
READY*
CLK
RESET
R-DISCRETE
VCC
300 Ohms
R1-C
5K
M23
XACK*
M14
INIT*
M25
LOCK*
BuffN1-A
74LS125
HOLD
BUS_ARBITER
LLCK
BREQ
BPRN
BPRO
BCLK
CBRQ
BUSY
INIT
SO/HLD
S1
M/IO
READY
CLK
RESET
LOCK
AL/CB
S/R
AEN
82289
R1-D
5K
PAL4
DACK3#
HF3
AF/E3
DACK2#
HF2
AF/E2
DACK1#
HF1
AF/E1
DACK0#
HFO
AF/EO
/dack3
hf3
afe3
/dack2
hf2
afe2
/dack1
hf1
afe1
/dack0
hfo
afeO
/ready
/mrdc
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
I12
I13
I14
O1
O2
O3
O4
O5
O6
O7
O8
nc
nc
dclk
nc
dreq3
dreq2
dreq1
dreq0
20L8
DCLK
DREQ3
DREQ2
DREQ1
DREQ0
BUS_CONTROLLER
INTA
IORC
IOWC
MRDC
MWTC
DT/R
DEN
ALE
MCE
SO
S1
M/IO
READY
CLK
CEN/AEN
CENL
CMDLY
MB
82288
AEN*
1,2
...active when
DMA has multibus
R1-E
5K
DFF1-A
74LS74
R1-F
5K
SOOUT
shift out output FIFO
nc
address_latch_strobe
data_transceiver_enable
data_transceiver_direction
ALE
DEN
DT/R*
MF0..92]
MULTIBUS[0..92]
Master Interface, Dreq Gen. & FIFO out En.
Size
A
Document Number
Image Convolution System
REV
3.0
Date: November 29, 1989
Sheet 3 of 11

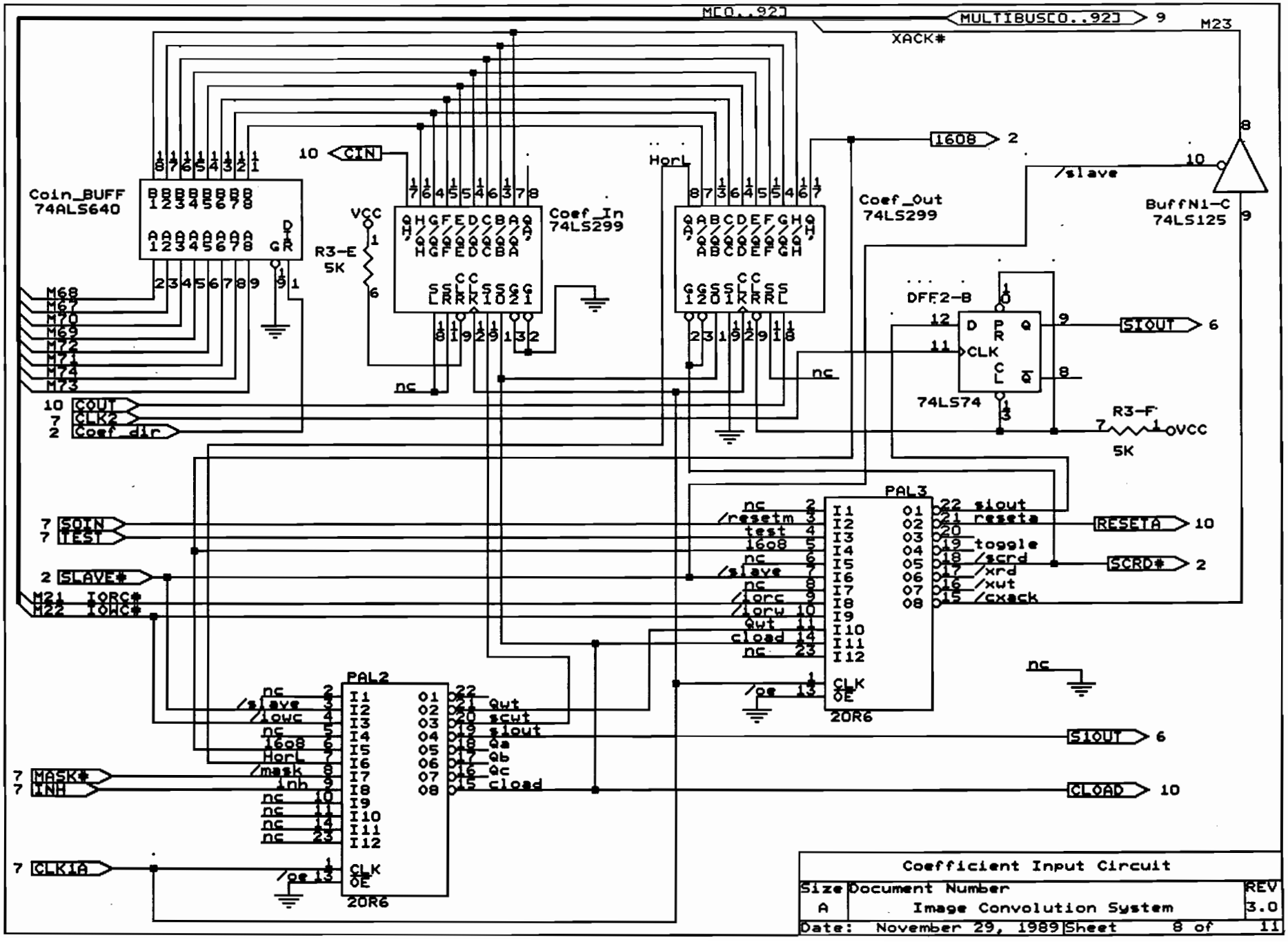

MD[0..92]
MULTIBUS[0..92]
R1-G
VCC
5K
XACK_COUNTER
XACKEN#
CLR
LOAD
CLK
ENT
ENP
RCO
D
C
B
A
QD
QC
QB
QA
74LS163
CLK
M23
XACK#
BuffN1-B
74LS125
INV1-A
74AS04
BREL
bus release
BuffP1-B
74LS126
ALE
address latch enable
BHE#
M27
BHEN#
BuffP1-C
74LS126
AEN
74LS74
DFF1-B
R1-H
5K
Data In
Latch/Mux1
74ALS533
Latch/Mux2
74ALS533
MD[0..7]
INPUT-D-BUS[0..7]
INV1-B
74AS04
MSBEN
DCLK
Slave XACK Gen, BHE# & IN/OUT FIFO Buffer
Size A
Document Number
Image Convolution System
REV 3.0
Date: November 29, 1989
Sheet 4 of 11

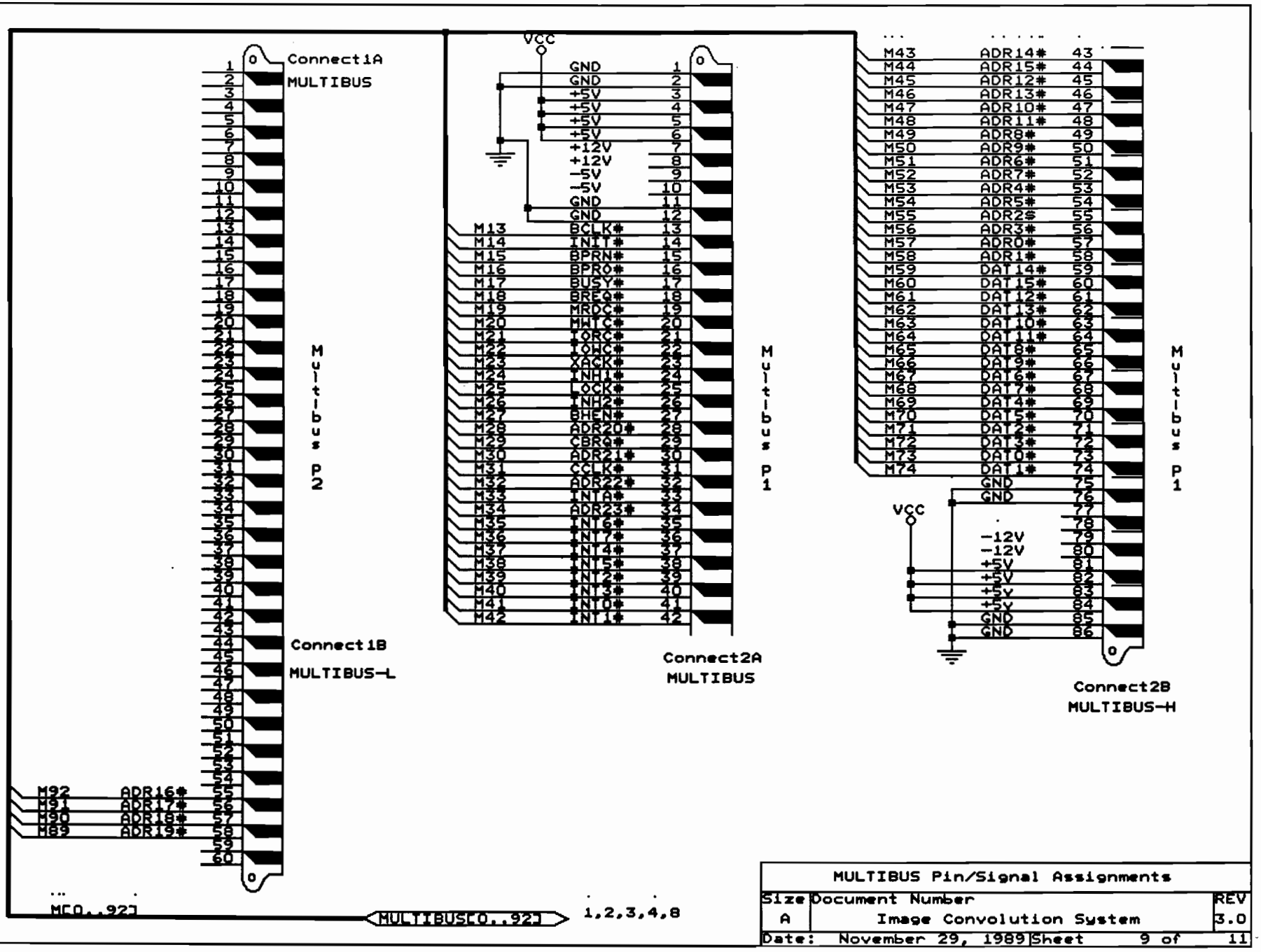

ROW I+0
ROW I+1
ROW I+2
PE
CLK1A
SFTIN1
74LS299
SFTIN2
74LS299
SFTIN3
74LS299
R3-G
5K
VCC
HF1
HF2
HF3
AF/E1
AF/E2
AF/E3
ORO
OR1
OR2
67413
FIFO_1H
FIFO_1L
FIFO_2H
FIFO_2L
FIFO_3H
FIFO_3L
SOIN
SIIN3
SIIN2
SIIN1
RESET+
DFO..7]
INPUT-D-BUSCO..7]
nc
Input Buffers
Size
A
Document Number
Image Convolution System
REV
3.0
Date: November 29, 1989
Sheet 5 of 11


8 SIOUT
7 CLK1A
VCC
5K
R3-A
10 YOUT
SFTOUT1
74LS299
SFTOUT2
74LS299
LOWBIT 2
2 STATUS
8 SIOUT
3 SOOUT
2 RESET+
1 DACK0*
FIFO_DHH
FIFO_DHL
FIFO_DLH
FIFO_DLL
67413
Local_bus[0..15] 1
IROUT 7
AFFO 3
HFO 3
Convolver and output FIFO
Size A
Document Number
Image Convolution System
REV 3.0
Date: November 29, 1989
Sheet 6 of 11

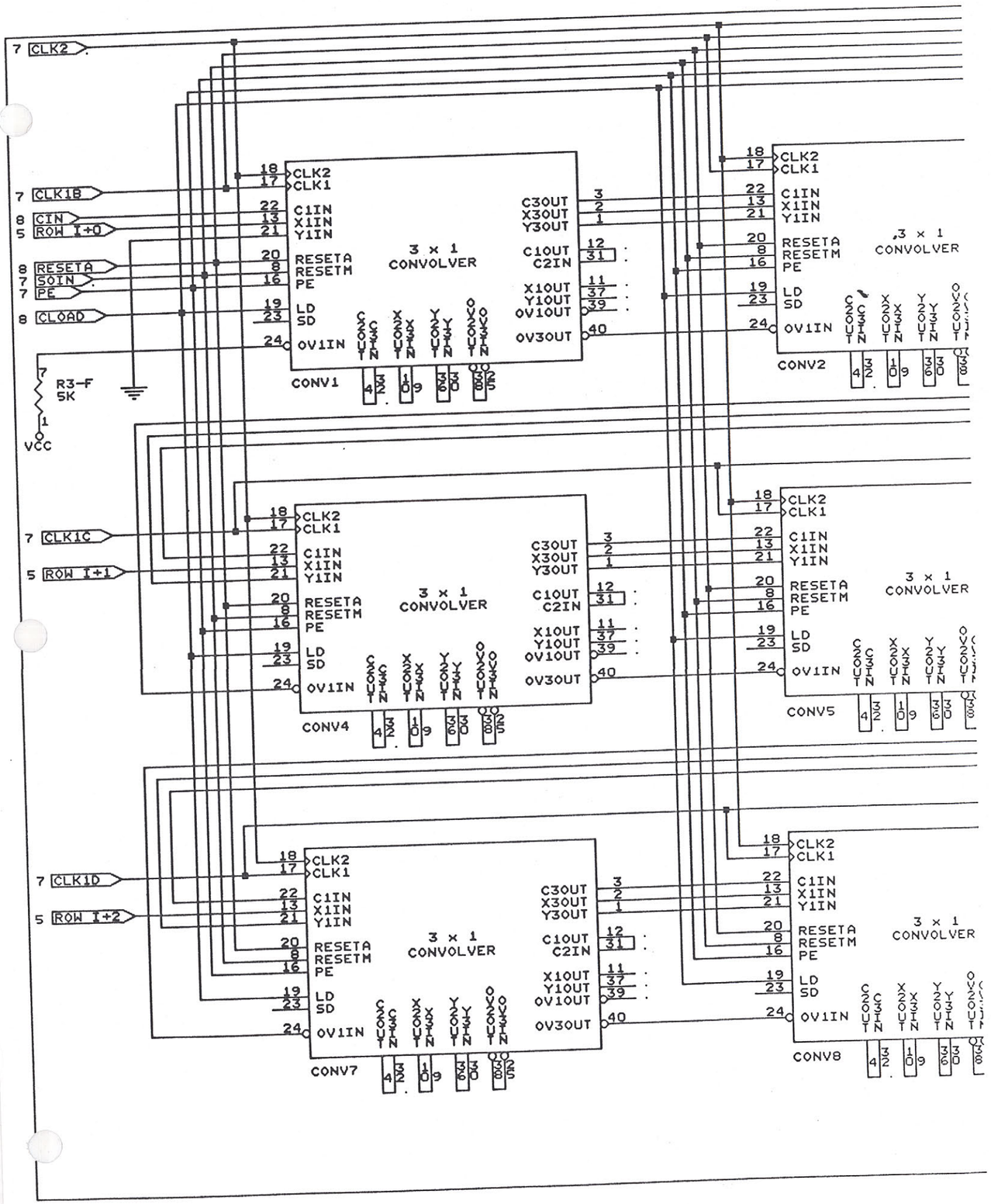


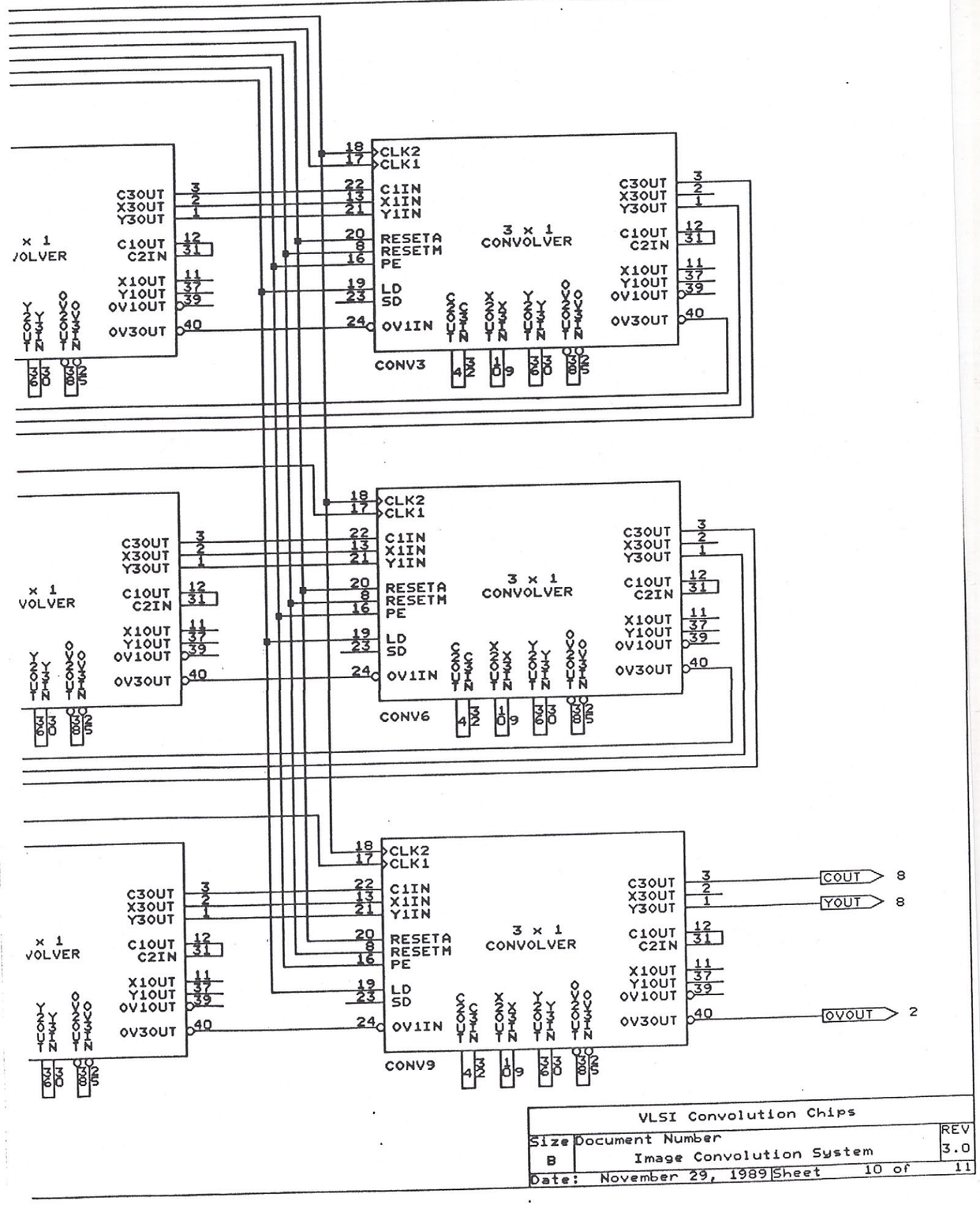
Clock Gen. #2 and Convolver Control		
Size A	Document Number Image Convolution System	REV 3.0
Date: November 29, 1989	Sheet 7 of 11	


MULTIBUS[0..92]
XACK#
M23
1608
Coef_Out
74LS299
Coef_In
74LS299
Coin_BUFF
74ALS640
BuffN1-C
74LS125
/slave
DFF2-B
74LS74
SIOUT
R3-E
5K
R3-F
5K
VCC
CIN
HorL
nc
COUT
CLK2
Coef_dir
SOIN
TEST
SLAVE#
IORC#
IOHC#
MASK#
INH
CLKIA
RESETA
SCRD#
CLOAD
PAL3
PAL2
20R6
Coefficient Input Circuit
Size A
Document Number
Image Convolution System
REV 3.0
Date: November 29, 1989
Sheet 8 of 11



MULTIBUS Pin/Signal Assignments		
Size A	Document Number Image Convolution System	REV 3.0
Date: November 29, 1989	Sheet 9 of 11	





VLSI Convolution Chips		
Size B	Document Number Image Convolution System	REV 3.0
Date: November 29, 1989	Sheet 10 of 11	

APPENDIX C: PAL's DESIGN STRATEGIES

C1. Description of PAL1:

Qa and Qb form a simple 4-bit counter. COND is used to determine if the convolver can process another set of inputs. On pulse 14, this condition is verified and INHIBIT is generated if the condition fails. This inhibit condition will be removed in a successive pulse 14 when COND is high again. A graphical representation of PAL1, its description and a timing diagram of these signals are as given in Table 9.1, Figure 9.1 and Figure 9.2.

```
CHIP PAL1 PAL20R6
clk  or0  or1   or2  inh  irout  nc  nc  nc  nc  nc  GND
/oe  nc  soin  Qf  Qe  Qd  Qc  Qb  Qa  cond  nc  Vcc
EQUATIONS
/Qd  :=  /Qd*/Qb  +  /Qd*/Qc  +  /Qd*/Qa  +  Qa*Qb*Qc*Qd
/Qc  :=  /Qc*/Qb  +  /Qc*/Qa  +  Qc*Qb*Qa
/Qb  :=  /Qa*/Qb  +  Qa*Qb
/soin  =  /Qf  +  inh
/Qa  :=  Qa
/Qe  :=  /Qa+Qd+/Qb+/Qc
/Qf  :=  /Qe
/cond  =  /or0  +  /or1  +  /or2  +  /irout
```

Table 9.1 PAL1 Description

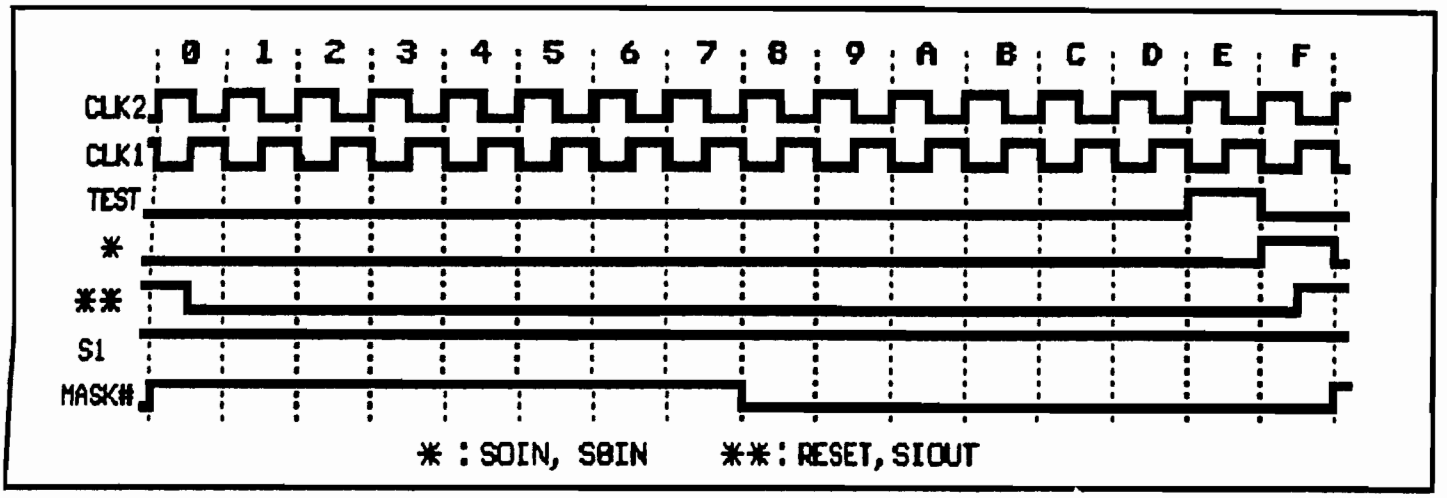


Figure 9.1 Timing Diagram for PAL1.

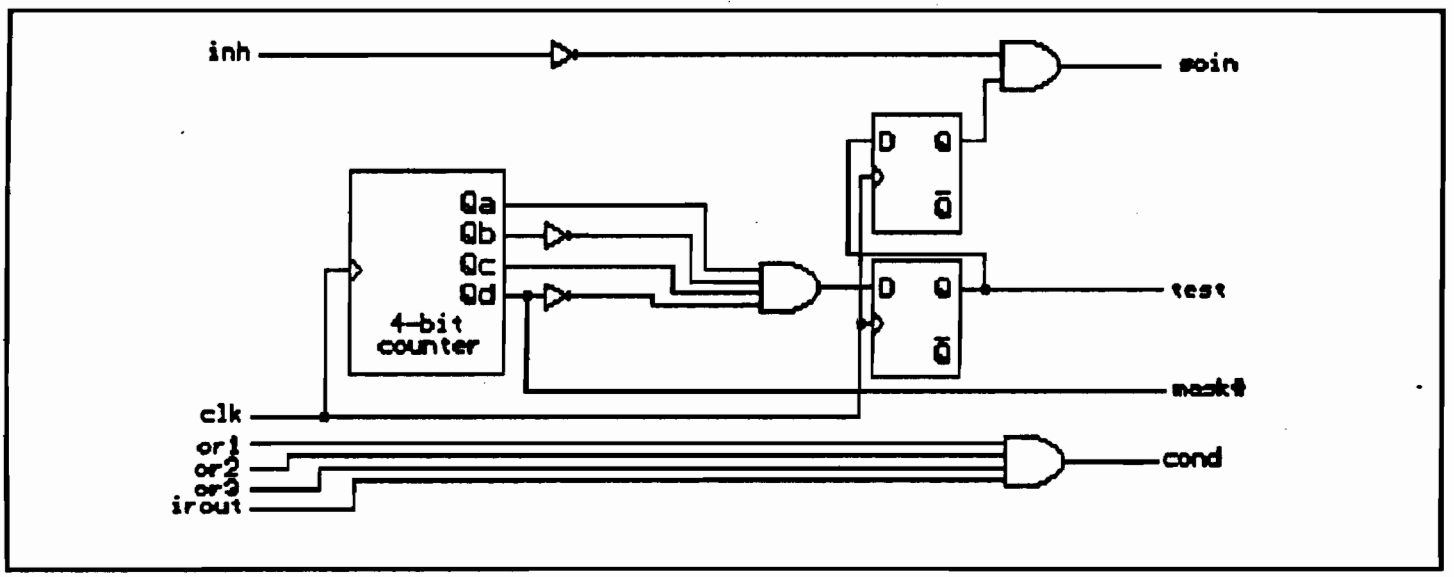


Figure 9.2 Logic Representation of PAL1.

C2. Description of PAL2:

PAL2 is a 3 bit self-starting counter that is used to shift the coefficients into the Convolver. When a coefficient is written into the shift register, PAL2 counts 8 cycles to shift

the 8 bits. Its description and graphical representation is given in Table 9.2 and Figure 9.3 respectively.

```
CHIP PAL2 PAL20R6
clk /low /high /iowc nc 16o8 HorL /mask inh nc nc GND
/oe nc cload Qc Qb Qa slout scwt Qwt shtleft nc Vcc
EQUATIONS
/Qc :=/Qc*/Qa + /Qb*/Qc + Qa*Qb*Qc
/Qb := Qb*Qa + /Qb*/Qa
/Qa := Qa + /scwt*/Qb*/Qc
/cload = /Qc * /scwt * /Qa * /Qb
/shtleft = Qc + Qb + Qa + scwt
/Qwt := /low + /high + /iowc
/scwt := /low + /high + /iowc + Qwt
/slout := inh + 16o8*/mask*HorL + 16o8*mask*/HorL
```

Table 9.2 PAL2 Description

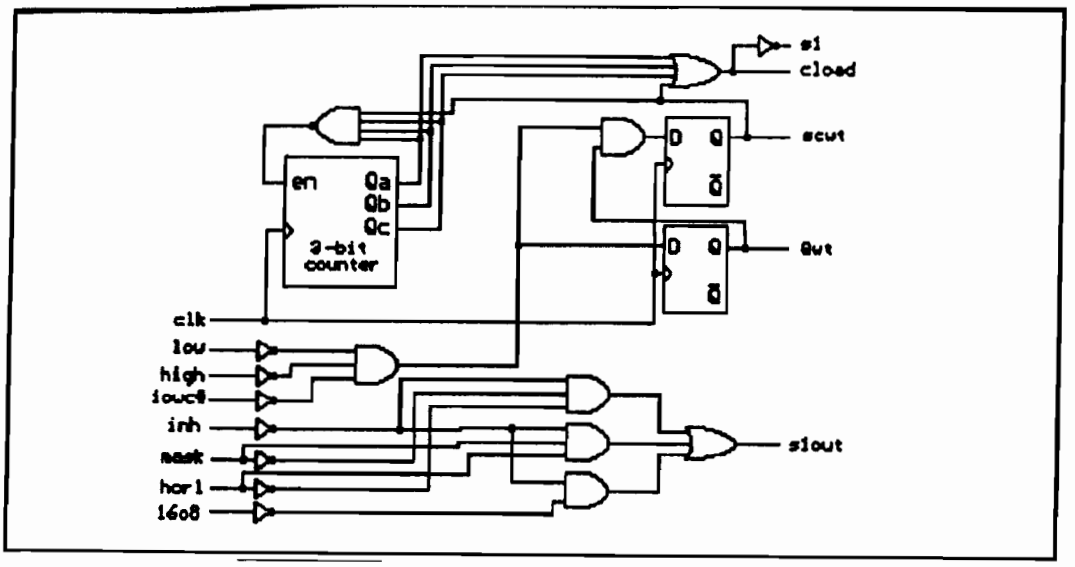


Figure 9.3 Logic Representation of PAL2.

C3. Description of PAL3:

PAL3 is used to generate the XACK signal when a coefficient is read or written from or into the Convolver. Toggle is a signal that alternates every 16 clock cycles of the Convolver and is used when bytes are to be saved as the resulting image. Refer to Table 9.3 for its description and Figure 9.4 for its graphical representation.

```
CHIP PAL3 PAL20R6
clk /Xack nc test 16o8 inh /low /high /iorc /iowc Qwt GND
/oe cload /cxack xwt xrd scrd toggle dqwt nc xen nc Vcc
EQUATIONS
/xen = /low + /high
/toggle := inh*/toggle + /toggle*/test*16o8 + toggle*test*/inh*16o8
/scrd := /iorc + /high + /low
/dqwt := /qwt
/xrd := scrd
/xwt := dqwt*/cload*iowt
cxack = /xwt + /xrd
```

Table 9.3 PAL3 Description

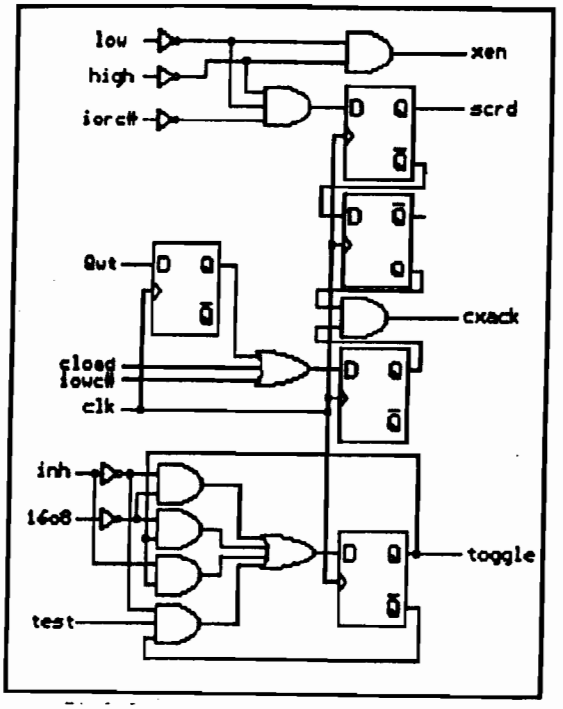


Figure 9.4 Logic Representation of PAL3.

C4. Description of PAL4:

PAL4 generates the DMA requests for the 4 channels using direct inputs from the FIFO's as described in Chapter 6. Refer to Table 9.4 for its description and Figure 9.5 for its graphical representation.

```
CHIP PAL4 Pal20L8
/dack3 hf3 afe3 /dack2 hf2 afe2 /dack1 hf1 afe1 /dack0 hf0 GND
afe0 reset nc resetp d0 nreset dreq3 dreq2 dreq1 dreq0 scrd Vcc
EQUATIONS
/dreq0 = /hf0*afe0 + /dack0*/afe0
/dreq1 = /hf1*afe1 + /dack1*/afe1
/dreq2 = /hf2*afe2 + /dack2*/afe2
/dreq3 = /hf3*afe3 + /dack3*/afe3
/nreset = reset
/resetp = reset + scrd
/d0 = /dack0
```

Table 9.4 PAL4 Description

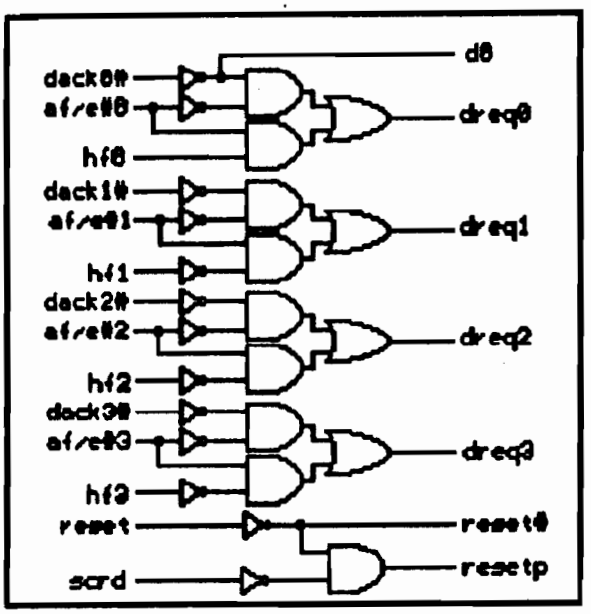


Figure 9.5 Logic Representation of PAL4.

C5. Description of PAL5:

Outbufen enables the output FIFO when the DMA is emptying it or when the CPU reads the contents of the coefficient shift register. Status is used as the low bit of the output FIFO. When the 16-bit mode is selected, it reflects if an overflow has been detected but leaves the low bit unaffected in the 8-bit mode. Refer to Table 9.5 for the PAL5 description and Figure 9.6 for its graphical representation.

```
CHIP PAL5 Pal20L8
/low /high /iorc  16o8 /dmahigh /aen dtr /dack0 /iowc lowbit den GND
brel ovout nc status nbrel /trans bd naen /cs outbufen nc Vcc
EQUATIONS
/nbrel = brel
trans = brel*den + brel*/aen + aen*den
/bd = iowc*/dtr*/dack0 + iowc*/aen + aen*/dtr*/dack0
/naen = /aen
cs = high*/aen
/outbufen = /low*/dack0 + /high*/dack0 + /iorc*/dack0
/status = 16o8*/ovout + /lowbit*/16o8
```

Table 9.5 PAL5 Description

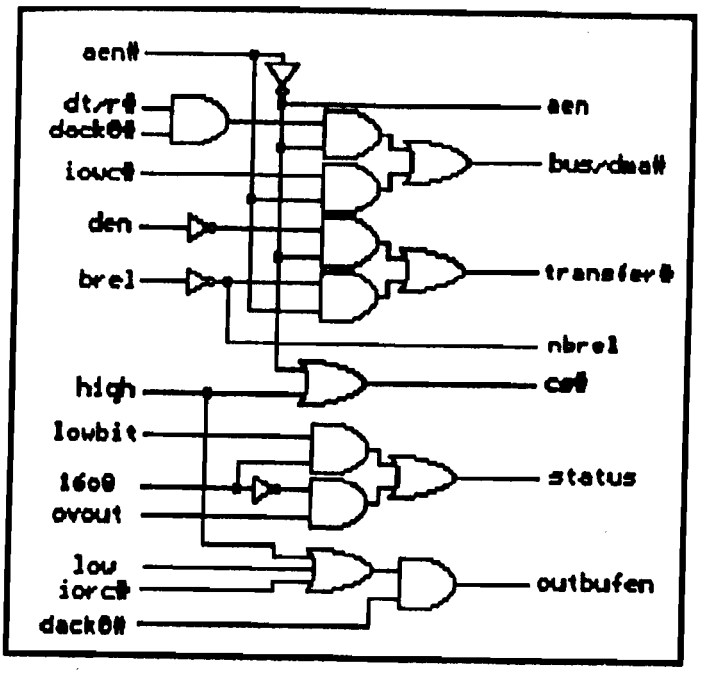


Figure 9.6 Logic Representation of PAL5.

C6. Description of PAL6:

PAL6 generates the controls when an input FIFO should grab data from the bus. The pulse synchronized signal Dclk goes high when XACK# is active. It generates the shift in (siin) signals since XACK# signifies that the data is valid on the bus. Siout is used to load data from the output shift register into the output FIFO. Refer to Table 9.6 for the PAL6 description and to Figure 9.7 for its graphical representation.

```
CHIP PAL6 Pal20R8
clk xack dack1 dack2 dack3 nc nc nc nc nc nc GND
/oe rst siout nc sinc inen siin3 siin2 siin1 dclk toggle Vcc
EQUATIONS
/Qdclk := xack + dack1*dack2*dack3
/dclk = xack + dack1*dack2*dack3 + Qdclk
/siin1 :=/dack1 + /inen*/dclk
/siin2 :=/dack2 + /inen*/dclk
/siin3 :=/dack3 + /inen*/dclk
/inen :=inen + /siin1*/siin2*/siin3
/siout = /toggle + /rst
```

Table 9.6 PAL6 Description

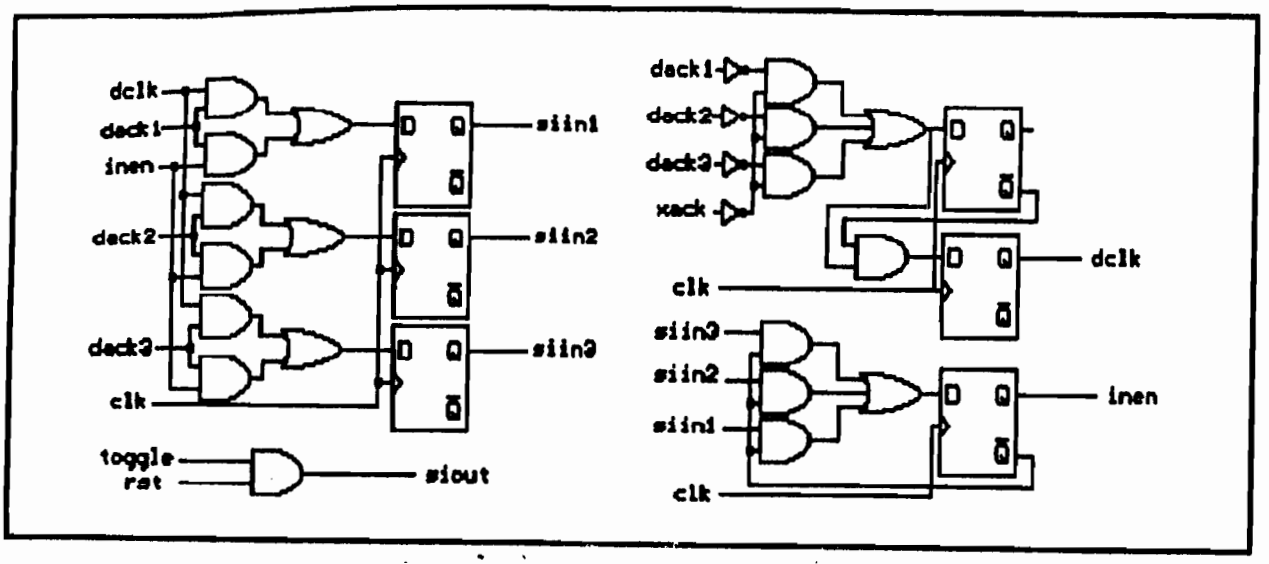


Figure 9.7 Logic Representation of PAL6.

C7. Description of PAL7:

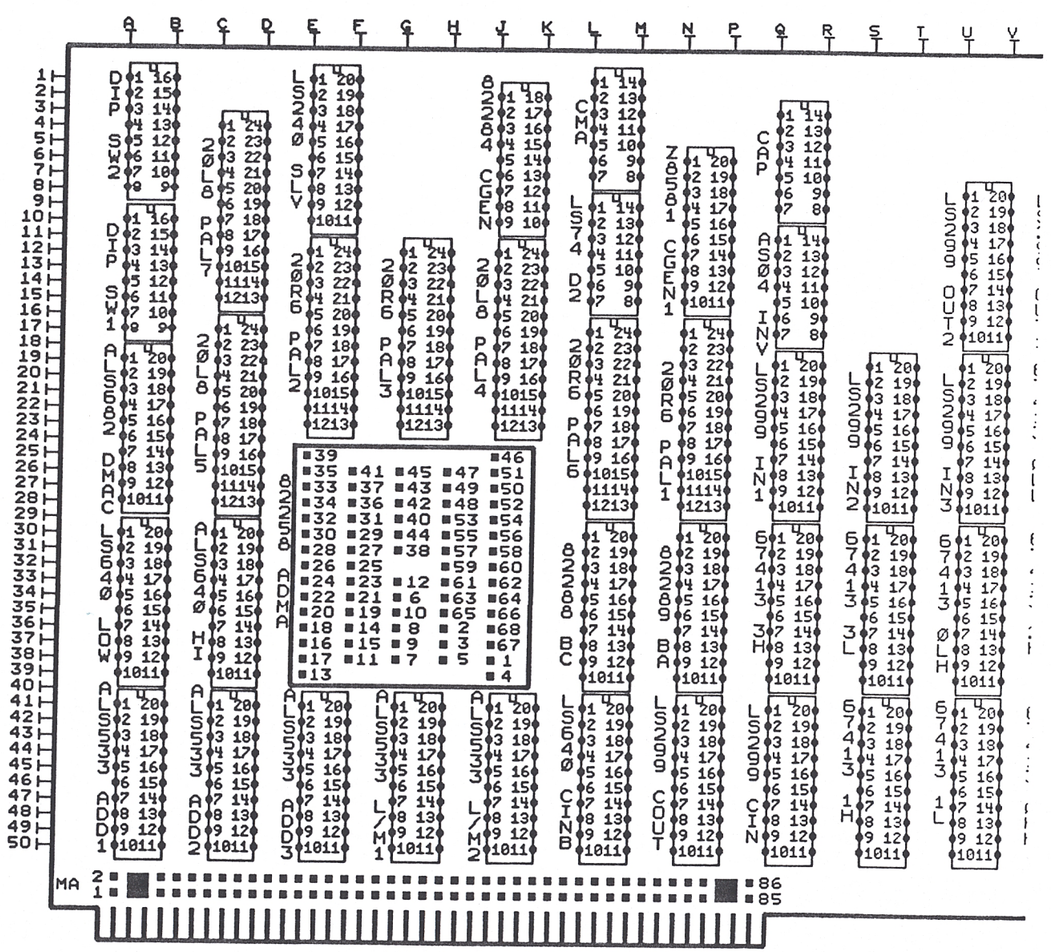
Control signals of PAL7 are covered in Chapter 5 and 6. Table 9.7 gives the description of PAL7.

```
CHIP PAL7 PAL20L8
 /DMAselect,/ADR0,/ADR1,/ADR2,/ADR3,/ADR4,/ADR5,/ADR6,/ADR7,reset,/scrd,GND
 /aen,/rrdack1,msben,/resetp,/ds,/SLAVE,/rrdack2,/rrdack3,dclk,siin1,/dack1,VCC

EQUATIONS

SLAVE = ADR7 * /ADR6 * /ADR5 * /ADR4 * /ADR3 * /ADR2 * /ADR1 * /ADR0 * DMAselect
ds = /SLAVE * DMAselect * /aen
/siin1 = /rrdack1 * /dclk + /rrdack1 * /dack1
/msben = /rrdack1 * /rrdack2 * /rrdack3
resetp = reset + scrd
```

Table 9.7 PAL7 Description

APPENDIX D: LAYOUT OF THE CONVOLUTION BOARD

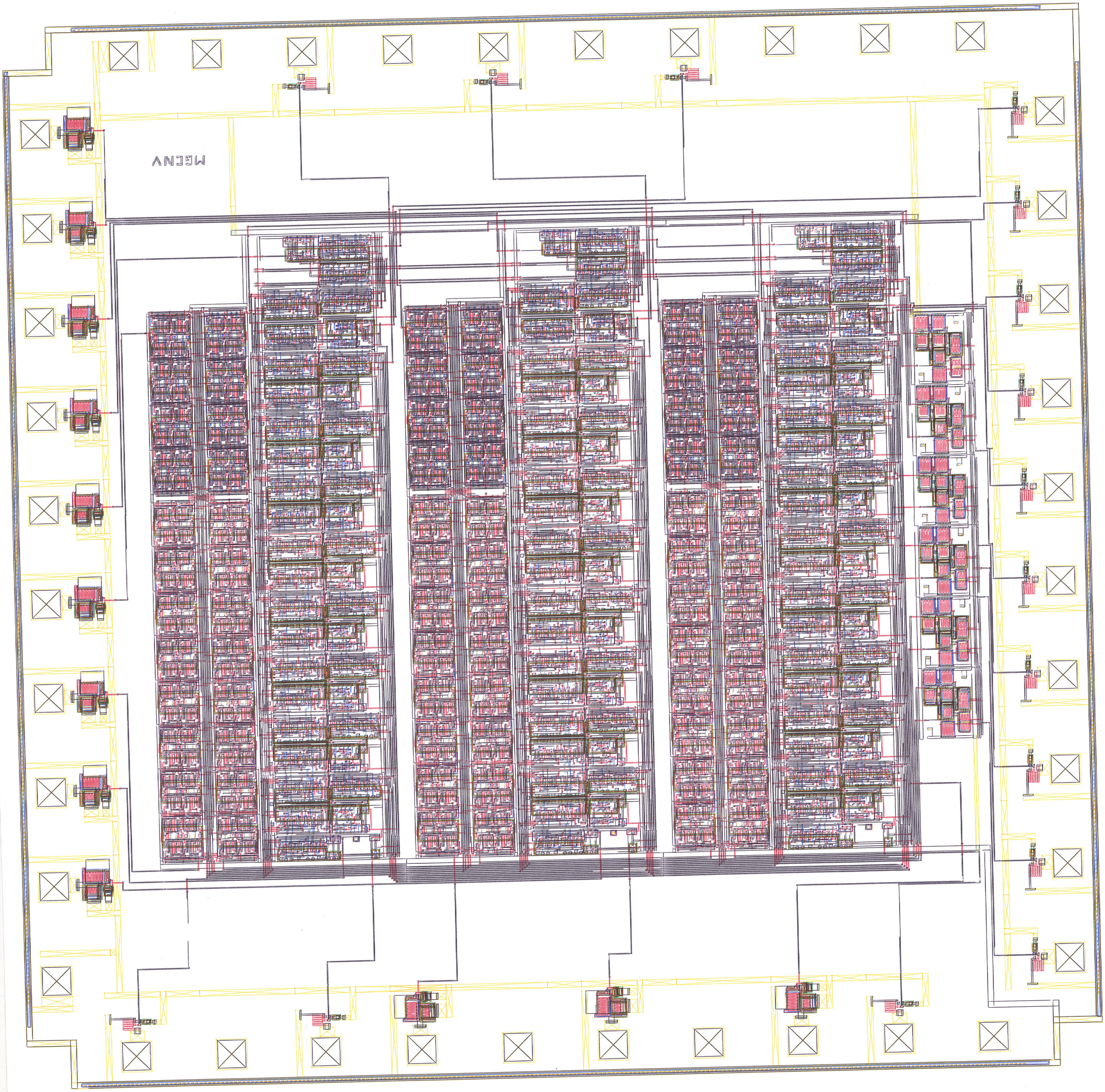
This page contains 1 Sheet



Board Layout		
Size: B	Document Number: Image Convolution System	REV: 3.0
Date: November 29, 1989	Sheet 11 of 11	

APPENDIX E: COLOR PLOT OF THE VLSI PROCESSOR

This page contains 1 Sheet


ANJSW