The Design and Construction of a Data Path Chip Set for a Fault Tolerant Parallel Processor

by

Charles E. Sakamaki

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fufillment of the requirements for the degrees of

**Master of Science in
Electrical Engineering and Computer Science**

and

**Bachelor of Science in
Electrical Science and Engineering**

at the

Massachusetts Institute of Technology

February 1991

Signature of Author ______________________________
Department of Electrical Engineering and Computer Science
February 1, 1991

Certified by ______________________________
Thomas F. Knight, Jr.
Thesis Supervisor

Certified by ______________________________
Richard E. Harper
Charles Stark Draper Laboratory

Accepted by ______________________________
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

The Design and Construction of a Data Path Chip Set for a Fault Tolerant Parallel Processor

by

Charles E. Sakamaki

Submitted to the Department of Electrical Engineering and Computer Science on February 6, 1991 in partial fulfillment of the requirements for the degrees of

Master of Science
and
Bachelor of Science

Abstract

Reliability of a computer architecture may be increased through fault tolerance. However, fault tolerance is achieved at a price of decreased throughput. The Fault Tolerant Parallel Processor at the Charles Stark Draper Laboratory maintains high levels of reliability and throughput by combining technologies of fault tolerance and parallel processing. The architecture is based on a Network Element (NE), which performs the functions of fault tolerance and parallel processing. A design for two field programmable gate arrays (FPGAs) is proposed herein which will replace much of the NE and perform the communication, synchronization, and redundancy management functions within the NE. This will yield increased reliability, reduced size, and reduced power dissipation. These FPGAs will be integrated with the next implementation of the Fault Tolerant Parallel Processor.

Thesis Supervisor: Thomas F. Knight
Title: Associate Professor of Electrical Engineering

Acknowledgments

This work would not have been possible without the expertise and support of the staff of the Fault Tolerant Systems Division of the Charles Stark Draper Laboratory. In particular, special thanks must be extended to Richard Harper, Bryan Butler, Martin McKinney, Mark Busa, Paul Mukai, and Eric Damon. I would also like to thank my thesis advisor Tom Knight for his support and direction, and finally my family and friends for their support and encouragement

This work was done at the Charles Stark Draper Laboratory under an internal research and development grant.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Problem Statement

The requirement of ultra-reliable computing has led to increasing research in the field of fault tolerant computing. Life critical and mission critical computing systems of the future will have performance and reliability requirements that the current fault tolerant computing systems cannot meet. These systems may be used in the decision-making processes of autonomous mechanisms, which may be used for both military and commercial purposes. An example of a military purpose would be a system to monitor incoming enemy missiles and destroy them when necessary. An example of a commercial purpose would be an automated air traffic controller.

Several fault tolerant computing systems have been designed according to the rules of Byzantine Resilience, such as the Fault Tolerant Multiprocessor (FTMP) [AH78], designed by the Charles Stark Draper Laboratory, and the Software Implemented Fault Tolerance (SIFT) Computer [JG84, JHW78, RWB86], designed by SRI International. However, the SIFT Computer proved inadequate for real-time operation due to the overhead resulting from the implementation of fault tolerance. The AIPS Fault Tolerant Computer (AIPS FTP) [AIPS84], designed by the Charles Stark Draper Laboratory, and the Network Element Based Fault Tolerant Processor (NEFTP) [TAA88], designed by the Charles Stark Draper Laboratory are more recent designs which can meet the performance requirements for real-time operations. The Charles Stark Draper Laboratory is in the process of designing a Cluster Three Fault Tolerant Parallel Processor (C3 FTPP), which surpasses previous fault tolerant systems regarding performance issues by combining properties of fault tolerance and parallel processing. Fault tolerance is necessary for high reliability and parallel processing enhances performance. The C3 FTPP requires a mechanism to provide fault detection and correction, synchronization, and inter-communication in a manner that is compact and reliable.

1.2 Objective

The primary objective of this thesis is to design and develop two field programmable gate arrays (FPGAs) which will be used for error detection/correction, synchronization, and communication purposes in the C3 FTPP. These FPGAs are named the Fault Tolerant Clock (FTC) FPGA and Voter FPGA, collectively known as the Data Path Chip Set. The FTC FPGA will handle the Network Element (NE) synchronization and inter-NE communication. The Voter FPGA will handle the error detection/correction. The FPGAs provide compactness, reliability, and low power dissipation for the C3 FTPP design. They provide a significant reduction in physical size over previous designs. The two sixty-eight pin FPGAs support the circuit equivalent of over forty TTL-type chip packages. Numerous design improvements result in increased reliability. CMOS FPGA technology allows for low power dissipation within the FPGAs, which also increases reliability. A design for these FPGAs is developed and fabricated.

1.3 Approach

This study begins with the discussion of some of the fundamentals of fault tolerant processing, in particular, the Byzantine Resilience approach. Several existing fault tolerant computers are then discussed, including the design of the C3 FTPP. The FPGA hardware is examined and the functional blocks for each FPGA are designed. The FPGAs are simulated and a general evaluation is made.

Chapter 2
Fault Tolerance Fundamentals

This chapter discusses some of the fundamentals of fault tolerance. The major motivation for fault tolerance is the need for ultra-reliable computing. As the demand for automated processes increases, those that are mission-critical and/or life-critical need ultra-reliable computing processes. For instance, a mission-critical operation might involve an unmanned spaceflight to another planet while a life-critical system might operate the life-support system of a space colony. Fault tolerant computing, as its name suggests, is computing that is able to tolerate faults in such a manner that errors can be detected and corrected prior to propagation throughout the system. Although no system can be 100% error-proof, a fault tolerant system can come close. In today's computer systems, fault tolerance is achieved through the use of redundancy. This redundancy can be done in either hardware or software, or both. An example of software redundancy would include multiple execution and comparison of the same software routines. Multiple processors executing the same software, or the same process in different manners is an example of hardware redundancy.

2.1 Fault Tolerance Requirements

Two key factors in comparing the fault tolerance of different computer architectures are performance and reliability. Performance is roughly measured by the throughput in millions of instructions per second (MIPS), while reliability can be measured as the probability of failure per hour. The throughput requirement on a fault tolerant system depends on the system's function. Most of today's real-time systems have throughput requirements ranging from one to ten MIPS. However, it is conceivable that a future system, such as an autonomous space mission, may have a throughput requirement of 100 MIPS. Typically, the allowable failure probabilities range from 10^{-4} to 10^{-6} per hour for mission-critical functions and 10^{-6} to 10^{-10} per hour for vehicle-critical and crew safety functions [JJD88].

There are two basic approaches to fault tolerance, the ad hoc approach and the Byzantine Resilience approach.

2.2 Ad Hoc Approach

The ad hoc approach provides fault tolerance for only those modes that are predicted to be likely to occur. This approach is taken in many commercial computers to provide minimal hardware redundancy. In the ad hoc approach, the probabilities of certain failure modes are estimated. Hardware is then designed to tolerate those modes judged to have a sufficiently high probability of occurrence. For instance, a system that operates in an area with faulty power lines may be designed with a backup power supply. Another example would be a data storage/retrieval system that stores multiple copies of data and uses checksums to retrieve correct data.

The ad-hoc approach is inadequate for ultra-reliable computing due to the fact that only the likely failure modes are tolerated, and because human assumptions can be unreliable. Even if the likely failure modes were completely tolerated, which itself is unlikely, the probability of failure is still the sum of the probabilities of all unlikely failure modes. For instance, the system with a backup power supply could easily be defeated by a faulty backup supply or any flaws the system itself has. The data storage/retrieval system could easily be defeated by a checksum that randomly matches corrupt data, or a faulty read/write mechanism.

2.3 Byzantine Resilience Approach

The Byzantine Resilience approach provides tolerance for all failure modes, including malicious failures, which are the worst possible failures that can occur. This approach is taken by several academic and research organizations. Although the probability of malicious failures cannot be determined, they must have a likelihood of less than 10^{-6} per hour to allow non-Byzantine resilient architectures to even have a chance of meeting the reliability requirements discussed in Section 2.1 The Byzantine Resilience approach is argued to yield much more reliable processes than the ad hoc approach since all failure modes are covered, not only those that are predicted.

The hardware cost of Byzantine resilient system is high, but is argued to be the only practical approach to ultra-reliable computing. The Byzantine Resilience approach is also much more beneficial than the ad-hoc approach since failure modes do not need to be estimated beforehand, thus eliminating the chance of error in human assumptions.

Chapter 3 Byzantine Resilience

Byzantine Resilience is derived from the consensus problem known as the Byzantine Generals Problem. As said in the previous chapter, it is the only practical approach to ultra-reliable computing. The Byzantine Generals Problem is summarized as follows. We imagine that several divisions of the Byzantine army, each commanded by its own general, are camped around an enemy city. The generals can only communicate with one another by messenger. After observing the enemy, the generals must decide upon a common action. However, some of the generals may be traitors, trying to prevent the loyal generals from reaching an agreement [LL82]. The generals must decide on whether to attack or retreat. The problem is to derive a protocol to ensure that all loyal generals agree on a common battle plan, obeying the commanding general whenever he is a loyal general.

The generals in the Byzantine Generals Problem represent processors in a computing system. Traitors represent processors that have malicious failures. The agreement on a common battle plan represents non-faulty processors agreeing on input data. By applying the solution to the Byzantine Generals Problem to a computer system we can develop a computing system that is operational even in the presence of malicious processor failures. This approach is used by the C3 FTPP, among others, to provide ultra-reliability. The communication process used by the Fault Tolerant Clock (FTC) FPGA and the voting process used by the Voter FPGA designed in this thesis is derived from the solution to the Byzantine Generals Problem.

3.1 Solution to the Byzantine Generals Problem

For a given number of traitorous generals, the solution to the Byzantine Generals Problem lies in the number of generals through which messages must be relayed, and to whom each general should relay the message. This solution guarantees that each loyal general will agree on a common battle plan, obeying the commanding general whenever he is a loyal general. Assuming deterministic unauthenticated protocols, a proven solution to the Byzantine Generals problem in the presence of f traitors (or f simultaneous faults) must meet the requirements shown in Table 3.1.

1. There must be at least 3f + 1 participants [LL82].
2. Each participant must be connected to at least 2f + 1 other participants through unique communication paths [DD82].
3. The protocol must consist of a minimum of f + 1 rounds of communication exchange among the participants to allow for the single sourcing of information [MJF82].
4. The participants must be synchronized to within a known finite skew [DD84-2].

Table 3.1 - Requirements for the Byzantine Generals Problem Solution

A system which meets the requirements shown in Table 3.1 is known to be f-fault Byzantine resilient. For example, a One-Byzantine resilient system must comprise at least four participants, each uniquely interconnected. It uses a two-round communication exchange, with each round of communication occurring during the same approximate time. A Two-Byzantine resilient system must comprise at least seven participants, each connected to at least five other participants. It uses a three-round communication exchange, with each round of communication occurring during the same approximate time.

3.2 The One-Byzantine Resilient System

The One-Byzantine resilient system must comprise four participants. This requirement can be depicted by showing that three participants do not allow for Byzantine fault tolerance.

Consider the case in which there are three participants, one commanding general and two lieutenants, each being able to relay the message of "attack" or "retreat". In the first scenario, Lieutenant 2 is a traitor, as in Figure 3.1. The Commander is loyal and sends the "attack" command to the two lieutenants. Lieutenant 2 then tells Lieutenant 1 that the Commander said to "retreat", since he is a traitor. Lieutenant 1 now cannot figure out what decision to make, since there is is conflicting data and a majority consensus cannot be met. [LL82]

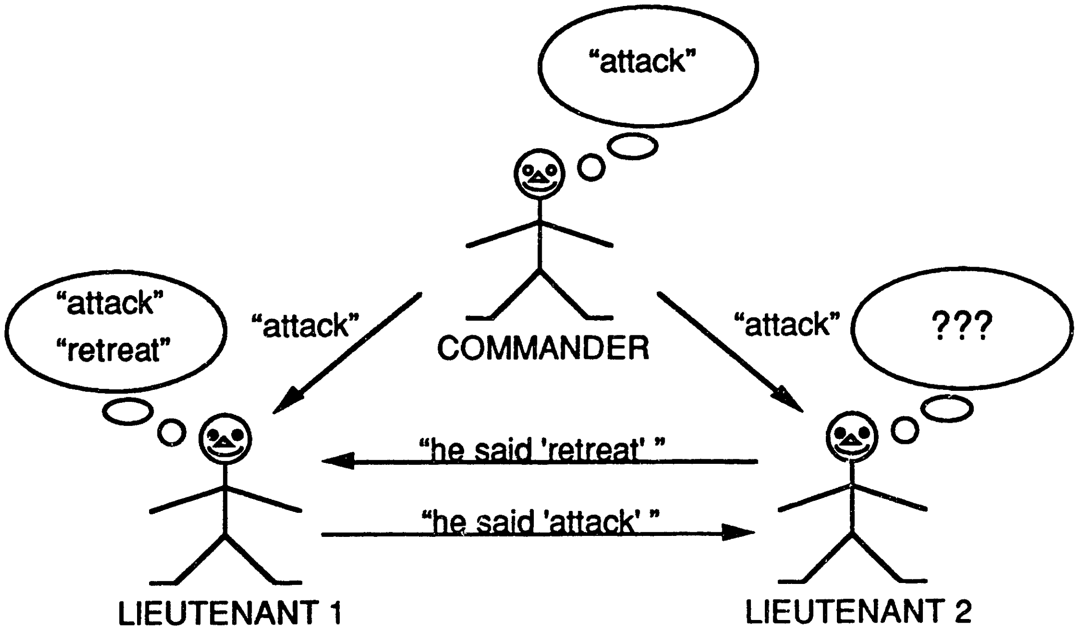


Figure 3.1 - Lieutenant 2 is a Traitor

In the second scenario, the Commander is a traitor, as in Figure 3.2. Since he is a traitor, the Commander sends the "attack" command to Lieutenant 1 and the "retreat" command to Lieutenant 2. Lieutenant 2 then tells Lieutenant 1 that the Commander said to "retreat". Lieutenant 1 cannot figure out what decision to make, since the information he has received is the same as in the first scenario. [LL82]

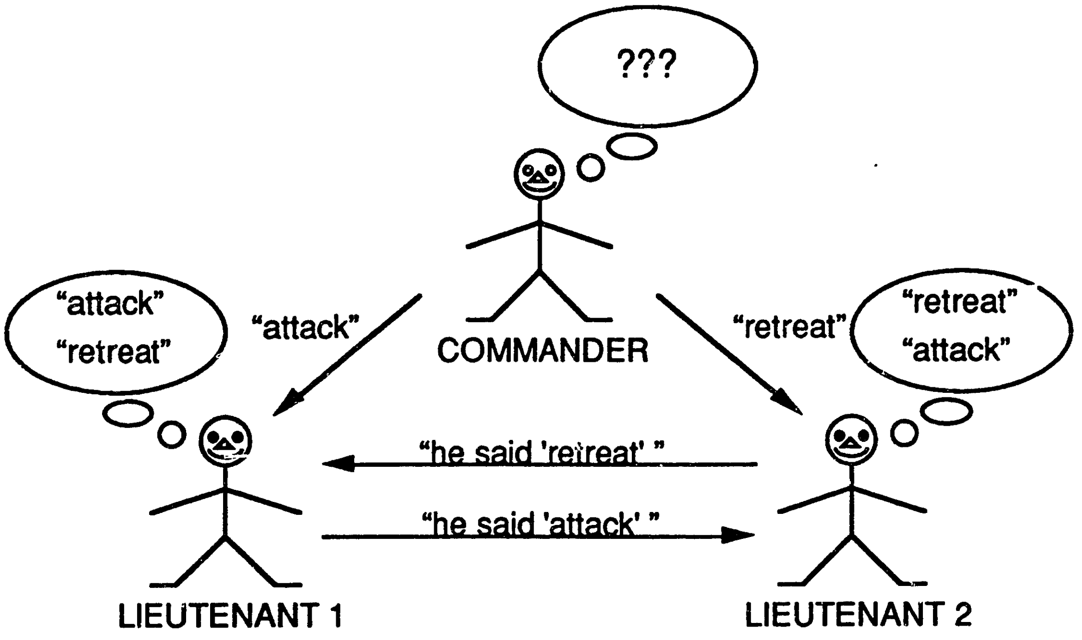


Figure 3.2 - Commander is a Traitor

With four participants, however, Byzantine resilience can be achieved even though there is one traitor. In the first scenario, Lieutenant 3 is a traitor, as indicated in Figure 3.3. The Commander is loyal and sends the message v to the three lieutenants, where v can mean either "attack" or "retreat". Lieutenants 1 and 2 then send the message v to the other Lieutenants, while the traitorous Lieutenant 3 sends a message y to Lieutenant 1 and a message x to Lieutenant 2. Lieutenants 1 and 2 can now do a majority vote on the orders received and carry out the message v. [LL82]

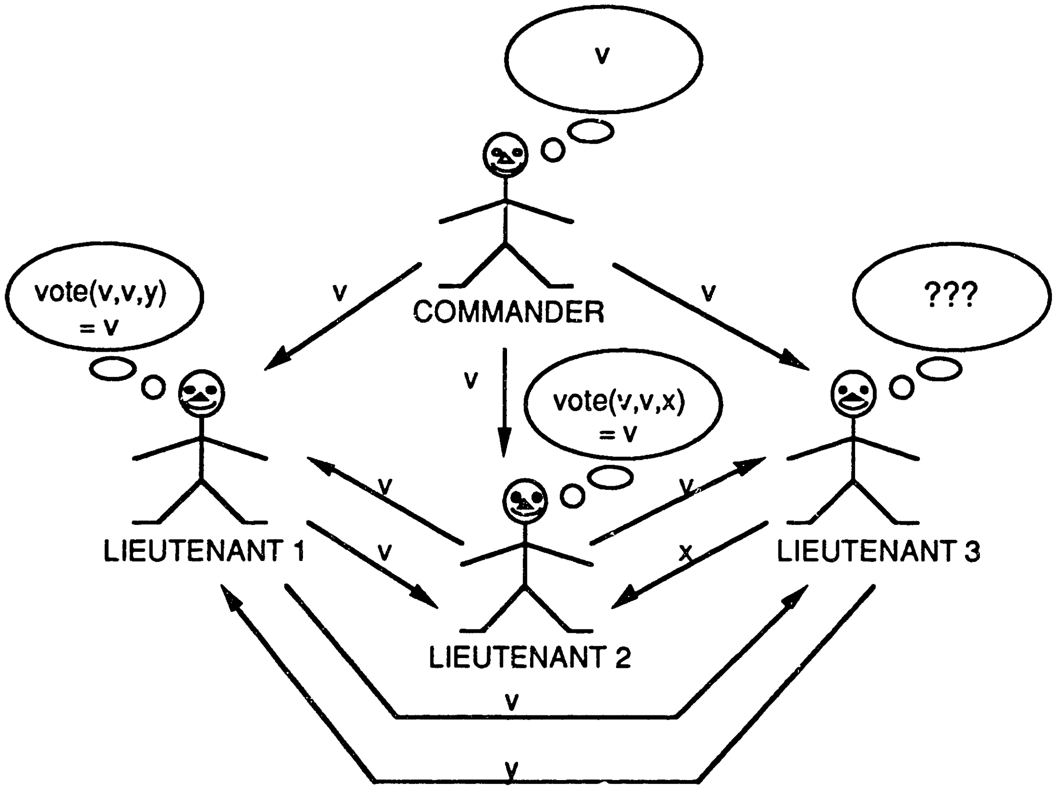


Figure 3.3 - Lieutenant 3 is a Traitor

In the second scenario, the Commander is a traitor, as indicated in Figure 3.4. The Commander sends the messages v, x, and y to Lieutenants 1, 2, and 3, respectively. The loyal Lieutenants, then relay the message to the other two Lieutenants so each Lieutenant receives one of each message. The Lieutenants now do a majority vote on the orders received and thus all receive the same message (the majority vote of v, x, and y), which we will call z. In either scenario, the loyal generals can determine which one is traitorous by comparing the majority vote of the orders with the ones they received. [LL82]

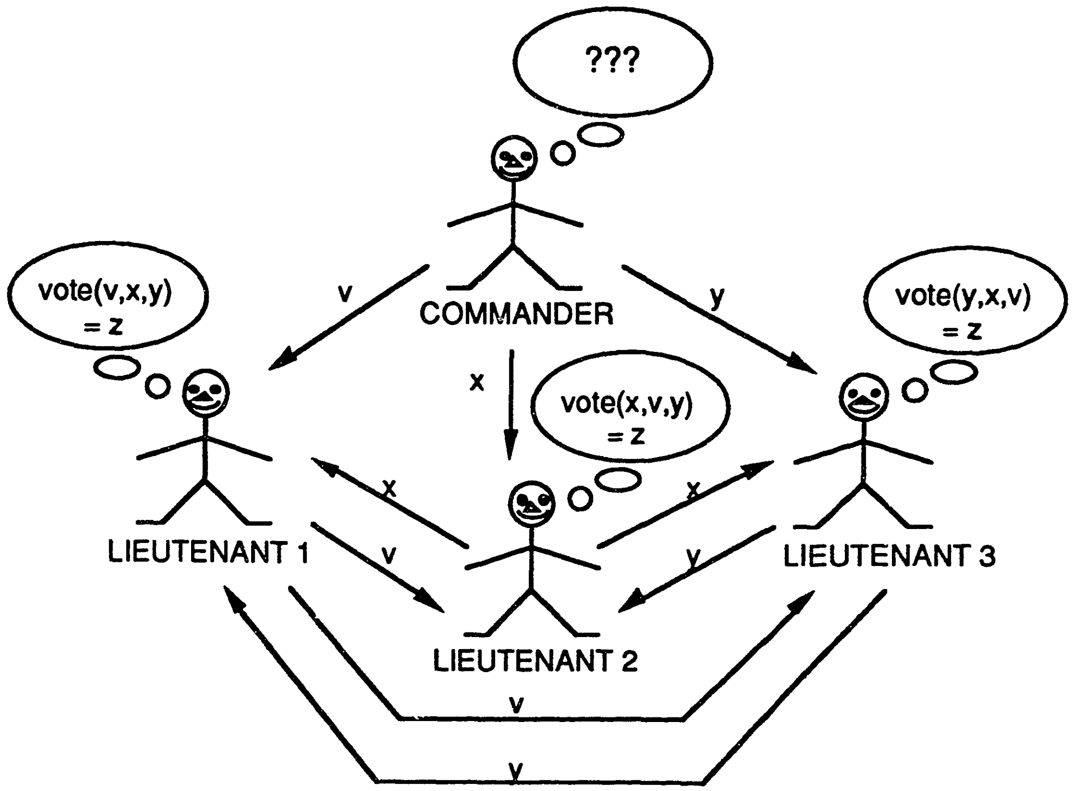


Figure 3.4 - Commander is a Traitor

So for single sourcing of data, we can create a One-Byzantine resilient system with four participants. A One-Byzantine resilient system consists of at least four Fault Containment Regions (FCRs). A One-Byzantine resilient system with four FCRs is shown in Figure 3.5.

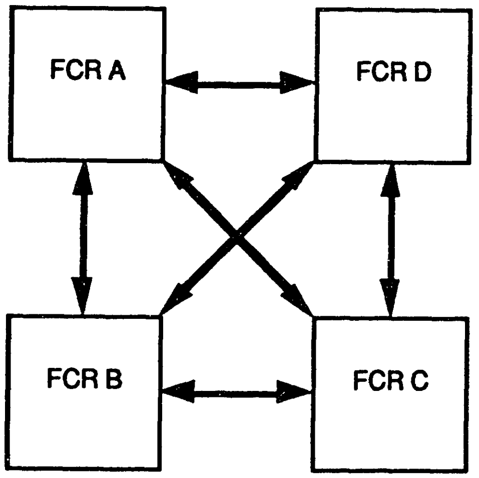


Figure 3.5 - A One-Byzantine Resilient System

Note that if there are at least three FCRs sourcing the same data in our four FCR system, only one round of communication needs to take place since a simple majority vote of incoming data may be used.

3.3 The Two-Byzantine Resilient System

For single sourcing of data, a Two-Byzantine resilient system must comprise at least seven FCRs, each connected to at least five other FCRs. It uses a three-round communication exchange, with each round of communication occurring during the same approximate time. A Two-Byzantine resilient system consisting of seven FCRs is shown in Figure 3.6.

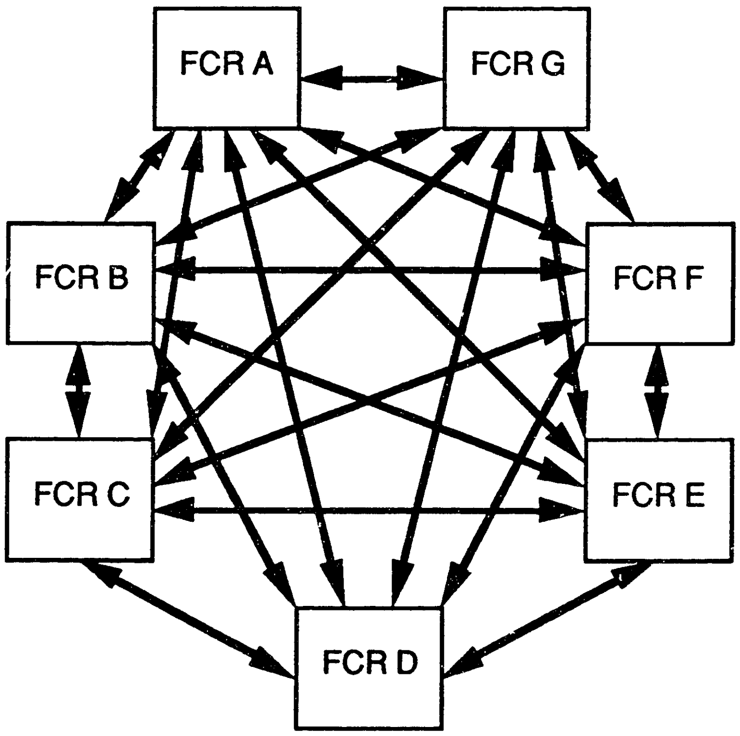


Figure 3.6 - A Two-Byzantine Resilient System

Note that if there are at least five FCRs sourcing the same data in our seven FCR system, only one round of communication needs to take place since a simple majority vote of incoming data may be used.

Chapter 4
Fault Tolerant Computers

There have been several architectures developed which satisfy the requirements for a One-Fault Byzantine resilient system. Among those discussed in this chapter are:

- The Software Implemented Fault Tolerance Computer (SIFT)
- The AIPS Fault Tolerant Computer (AIPS FTP)
- The Multicomputer Architecture for Fault Tolerance (MAFT)
- The Fault Tolerant Parallel Processor (FTPP)
- The Network Element Based Fault Tolerant Processor (NEFTP)

As the solution in Section 3.1 suggests, a One-Fault Byzantine resilient system must fulfill the following requirements.

1. There must be at least four participants [LL82].
2. Each participant must be connected to at least three other participants through unique communication paths [DD82].
3. The protocol must consist of a minimum of two rounds of communication exchange among the participants to allow for the single sourcing of information [MJF82].
4. The participants must be synchronized to within a known finite skew [DD84-2].

For simplicity of reference purposes, we will refer to requirement one as the *processor number requirement.* Requirement two will be referred to as the *connectivity requirement.* Requirement three will be referred to as the *communication requirement.* Finally, requirement four will be referred to as the *synchronization requirement.*

4.1 The AIPS Fault Tolerant Computer (AIPS FTP)

The AIPS Fault Tolerant Processor (AIPS FTP) [AIPS84] was developed at the Charles Stark Draper Laboratory, Inc. It consists of three processors and three interstages which act as Fault Containment Regions (FCRs) connected as shown in Figure 4.1. The interstages are simple data routers that forward the input to each of the processors. The processors are identical processors executing the same software. They resolve the input data by a 3-way, bit-for-bit majority voter circuit.

Synchronization is achieved through a clock signal called the Fault Tolerant Clock (FTC). This is employed by a digital phase lock loop circuit in each processor to lock in phase the FTC signal. The local FTC (LFTC) is approximately an 888 kHz clock signal. Each processor sends its respective LFTC to the interstages, which in turn relay the signals back to the processors. The processors generate FTC by selecting the median value of the three input signals, which should be the same for all processors as long as the interconnection skews between the processors and interstages are the same. All of the processors receive the same number of processor clock (SYSCLK) ticks during the FTC cycle.

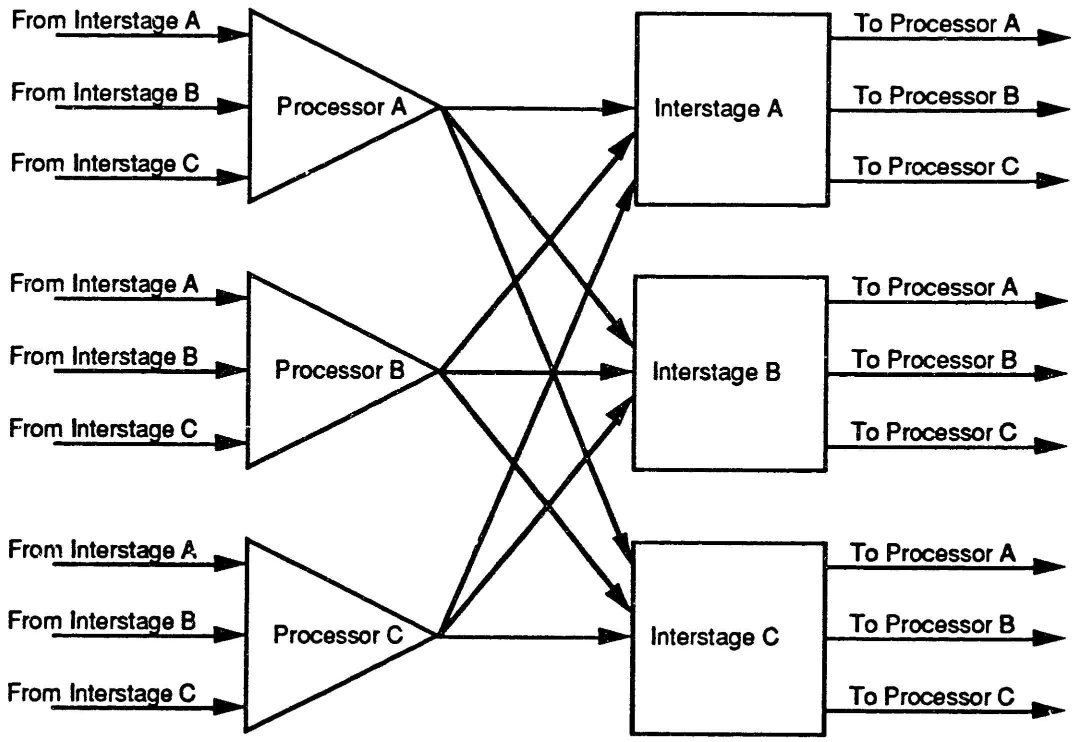


Figure 4.1 - AIPS Fault Tolerant Processor Architecture

The three processors of the AIPS FTP are each participants in the Byzantine Generals problem. The interstages act as the fourth participant, fulfilling the participant number requirement for Byzantine resilience. The connectivity requirement is fulfilled by the fact that each processor is connected to all of the interstages, and the interstages are all connected to each processor. Two rounds of exchange take place during the input consistency algorithm. In the first round, the processors forward their respective output data to each of the interstages. In the second round, the interstages forward the data they have received to each processor. These two rounds of communication fulfill the communication requirement for Byzantine resilience. Synchronization is achieved through the Fault Tolerant Clock as previously mentioned. The post-synchronization skew is held to within 125 ns, fulfilling the synchronization requirement.

4.2 The C1 Fault Tolerant Parallel Processor (FTPP)

The Cluster One (C1) Fault Tolerant Parallel Processor (FTPP) [JJD88, REH85, REH87-2] is a computer developed at the Charles Stark Draper Laboratory, Inc., which combines the principles of Byzantine resilience with parallel processing.

The C1 FTPP is grouped into at least four Fault Containment Regions (FCRs), each consisting of four Processing Elements (PEs) and a Network Element (NE) arranged as in Figure 4.2. Each FCR in the FTPP is physically isolated, electrically isolated (including separate power supplies), and clocked independently. The PEs are off-the-shelf, commercial processor boards that are VME compatible. The C1 FTPP at the Draper Laboratory uses 12.5 MHz versions of the Motorola 68020 VME boards as PEs.

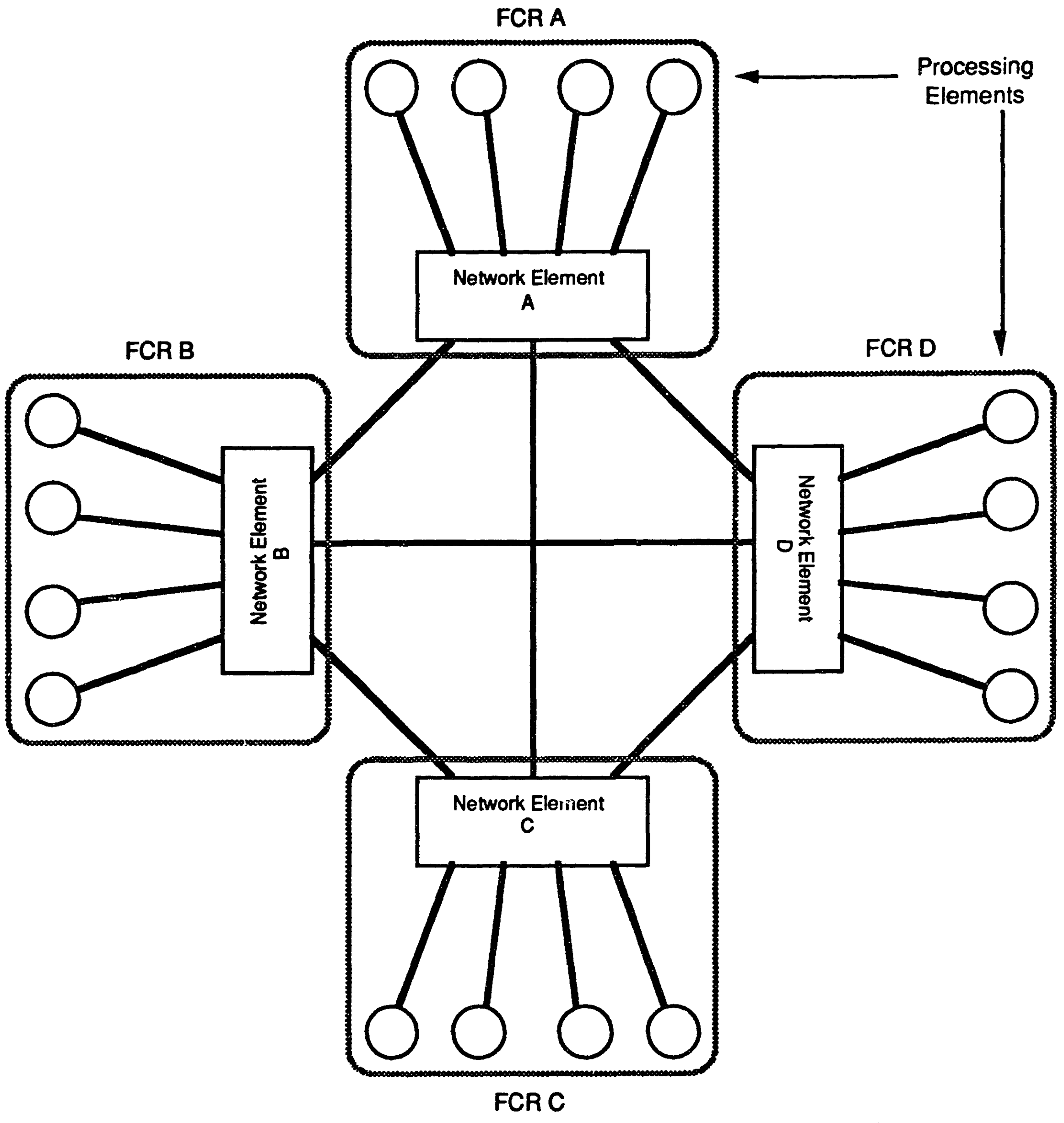


Figure 4.2 - The C1 Fault Tolerant Parallel Processor Architecture

The NE is the heart of the FTPP and performs the communication, voting, and synchronization functions of the system. The NE can be divided into six functional blocks:

- The PE/NE Interface
- The NE Data Paths
- The Inter-FCR Communication Links
- The NE Fault Tolerant Clock
- The NE Scoreboard
- The NE Global Controller

The PE/NE Interface provides for the passing of variable size messages between the PE and the NE. The NE Data Paths provides for the handling of one and two round communications exchanges, and the resolution of redundant data copies (majority voting). The Inter-FCR Communication Links provide for passing of data between the FCRs. The NE Fault Tolerant Clock synchronizes an NE with respect to the other NEs. The NE Scoreboard decides which processor exchange, if any, can be performed. Finally, the NE Global Controller is the central micro-controller for the NE. The block diagram of the C1 FTPP is shown in Figure 4.3.

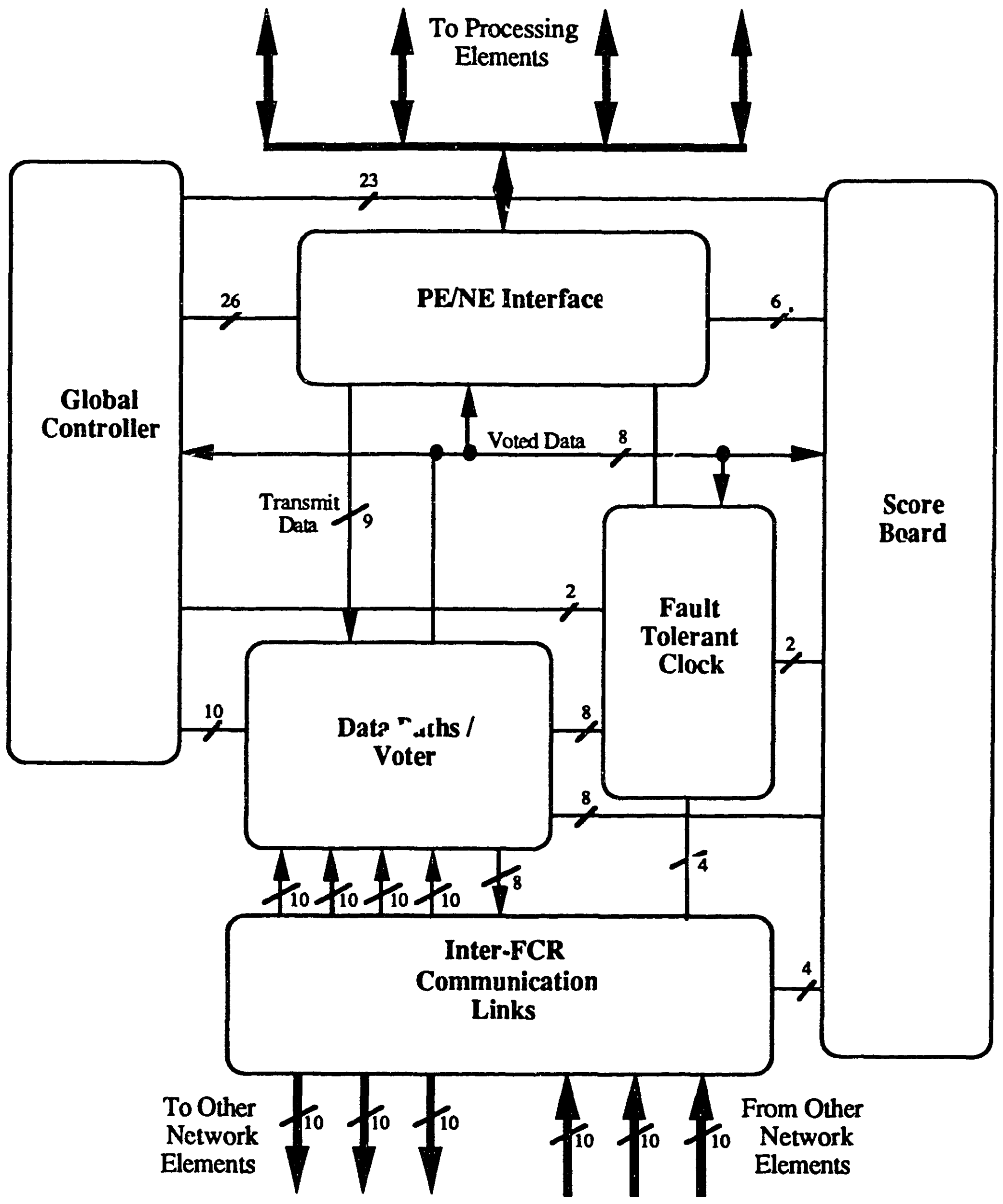


Figure 4.3 - Block Diagram of the C1 FTPP Network Element

The processors of the C1 FTPP are arranged as virtual processors, each consisting of one, three, or four physical processors. A virtual processor consisting of one, three, or four physical processors is called a simplex, triplex, or quadruplex virtual processor, respectively. This provides for different levels of fault tolerance and throughput. Each of the processors in the virtual processing group executes the same code. The members of a virtual processor comprise a Fault Masking Group (FMG). Each of the redundant processors in a virtual processing group reside in a separate FCRs to eliminate single point failures. Communication takes place between the processors by means of the NE.

To maintain synchronization and consistency, the processors exchange messages with each other. Each NE tells the other NEs whether or not its processor needs to exchange information. If all non-faulty processors request the exchange, then the exchange is performed. The data sent by a NE requesting an exchange of information is called an exchange request pattern. Four of these exchanges occur during the first half of this cycle. The series of these four exchanges is called a system exchange request pattern (SERP) exchange.

Once it is determined through a SERP that an exchange is to take place, there are two types of exchanges that can take place. The first type is a Class 1 exchange which is sourced by the members of a triplex or quadruplex virtual processor. During this type of exchange, each FMG member delivers its message to its respective NE, which then relays the message to the other three NEs. Each NE transmits one message and receives three messages. Note that all NEs transmit messages whether the virtual processor is running in triplex or quadruplex mode. When in triplex mode, the NE not hosting a member of the FMG delivers an empty message to the other NEs. The information received by each NE is then majority voted. Disagreements in the vote are relayed to the processing elements so that Fault Detection Isolation and Reconfiguration (FDIR) software can determine if an element is faulty and reconfigure the system as necessary. The Class 1 message exchange is diagrammed in Figure 4.4.

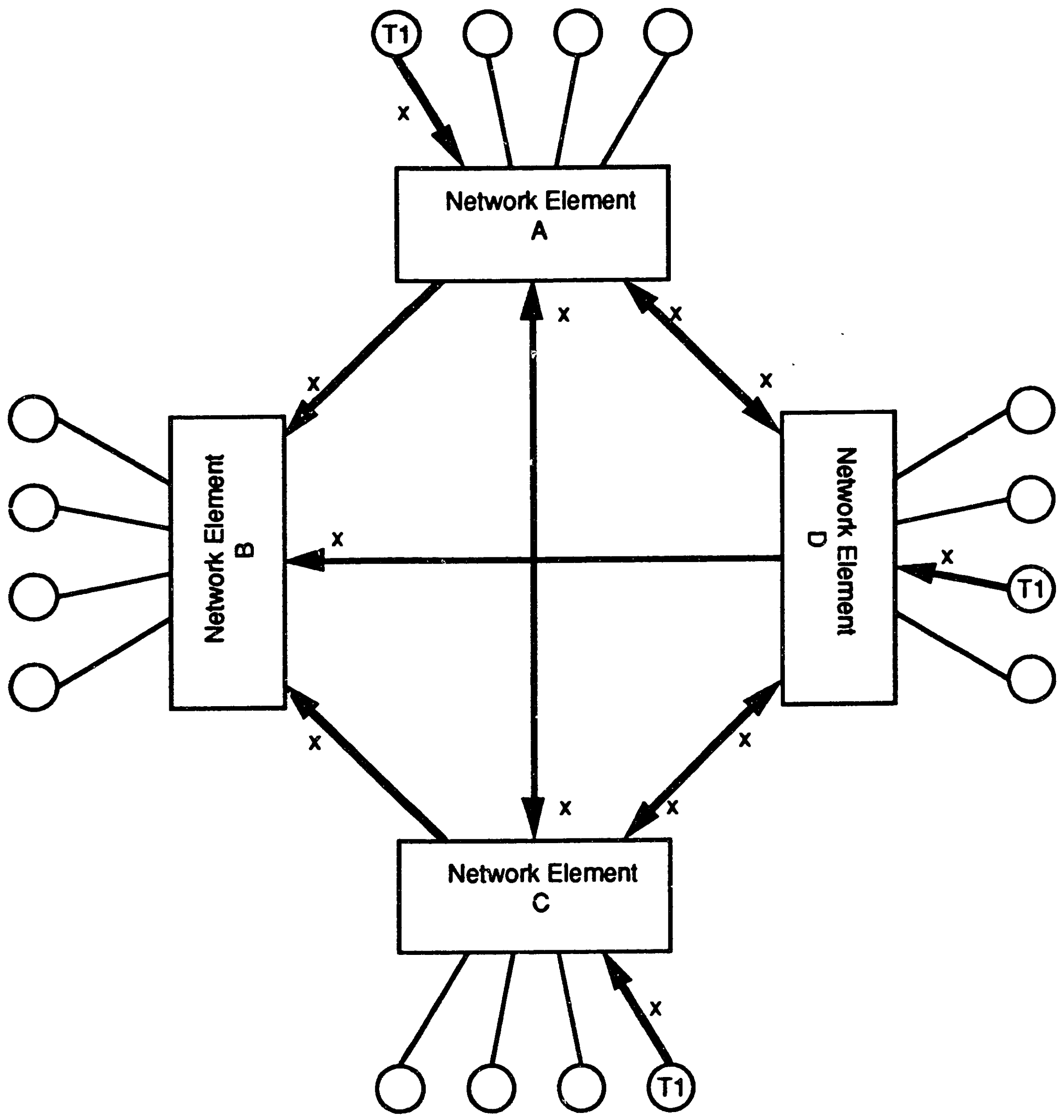


Figure 4.4 - C1 FTPP Class 1 Message Exchange

For a simplex virtual processor, a Class 2 message exchange is used. A Class 2 exchange is a two-round exchange. This is identical in concept to the two-round exchange diagrammed in Figures 3.3 and 3.4. In the first round, a member of the FMG single-sources a message to its hosting NE, which then relays this message to the other three NEs. In the second round, each NE reflects the message they received to the other three NEs. Thus, each NE has four copies of the message and can be majority voted. The Class 2 message exchange is diagrammed in Figures 4.5 and 4.6.

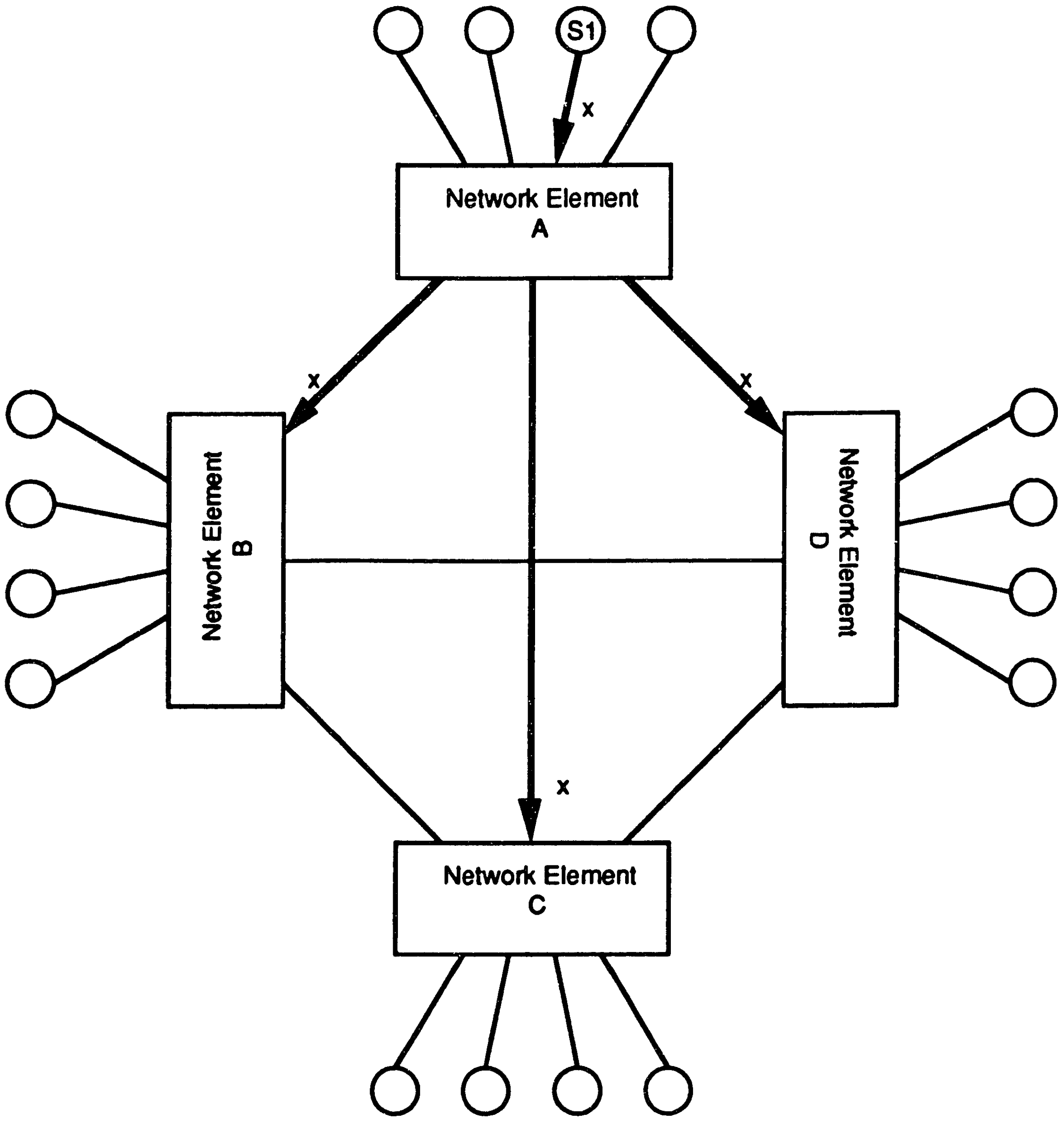


Figure 4.5 - C1 FTPP Class 2 Message Exchange (Round 1)

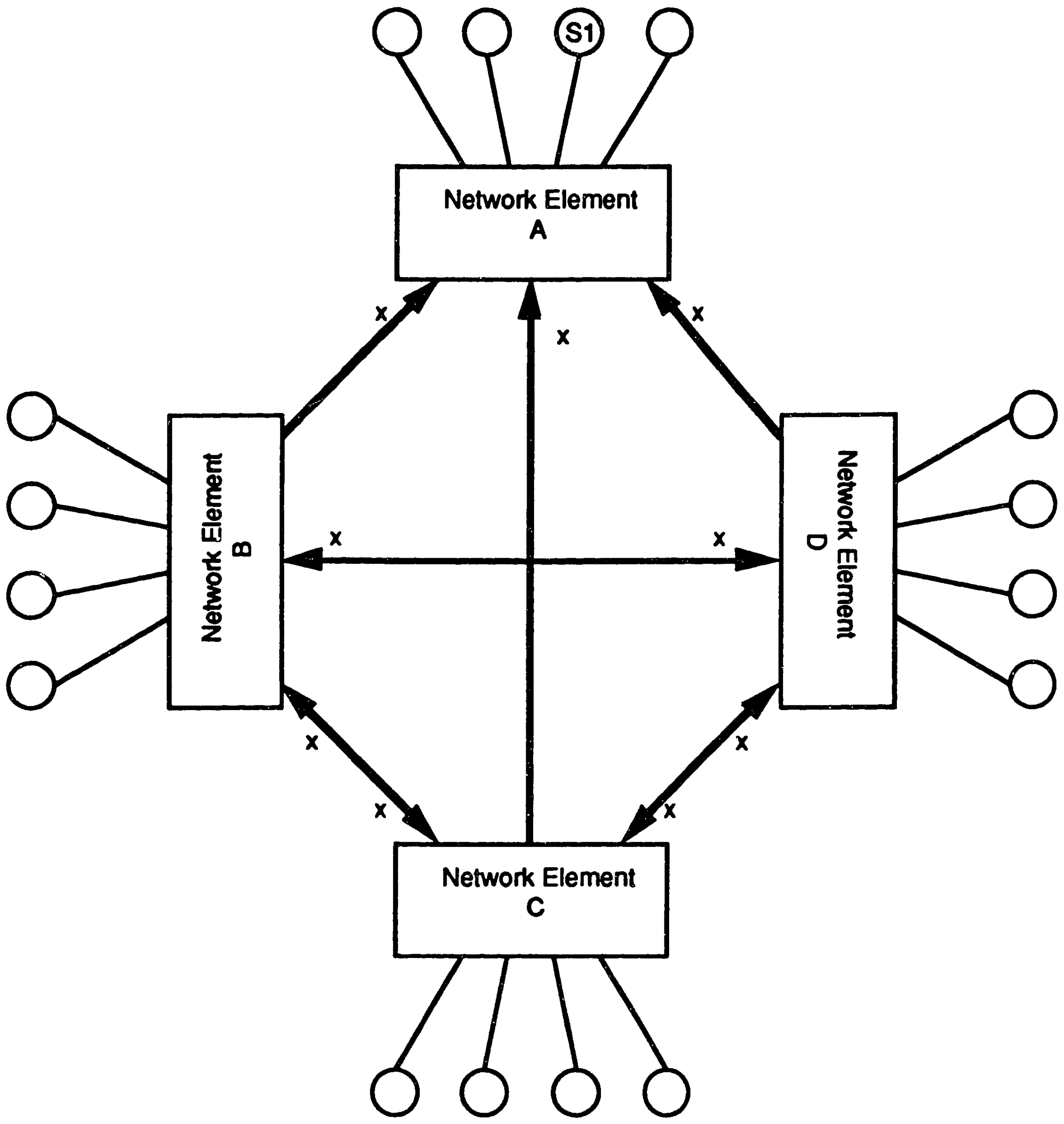


Figure 4.6 - C1 FTPP Class 2 Message Exchange (Round 2)

Since there are four NEs in the C1 FTPP, there are the equivalent of four participants in the Byzantine Generals Problem. This fulfills the participant number requirement for Byzantine resilience. Each NE is connected to the other three NEs via a metallic link. Thus, it meets the connectivity requirement. A Class 2 exchange of data is used during the simplex mode of operation. It is equivalent to two rounds of communication. The first round, involves each of the PEs completing a segment of code and sending its respective output data to the NE. This is similar to the way the commander issues commands to each of the lieutenants in the Byzantine Generals Problem. The second round involves the NEs reflecting the data to each of the other NEs and taking a majority vote of all the data. This is similar to the lieutenants' relaying of messages to the other lieutenants. The triplex and quadruplex also can perform Class 2 exchange modes which meet the communication requirements for Byzantine resilience. The NEs are synchronized by the Fault Tolerant Clock mechanism. When data is received outside of a limited time frame then an error is noted and the local NE's clock is adjusted accordingly. This method of keeping the NEs within a known skew fulfills the synchronization requirement. In triplex or quadruplex modes, data is simply majority voted after one round of exchange. This allows one fault to be detected and corrected during each round of exchange. Therefore, the C1 FTPP, meets all the requirements for being One-Fault Byzantine resilient.

4.3 The Multicomputer Architecture for Fault Tolerance (MAFT)

The Multicomputer Architecture for Fault Tolerance (MAFT) [AMF88] attempts to reduce the overhead of fault tolerance for increased throughput. It consists of four Operations Controllers (OCs) and four Applications Processors (APs), as shown in Figure 4.7. The OC is the device that handles the fault tolerance. It communicates back and forth to the AP and to the other OCs. It also performs the data voting, error detection, synchronization, reconfiguration, and task scheduling for the system.

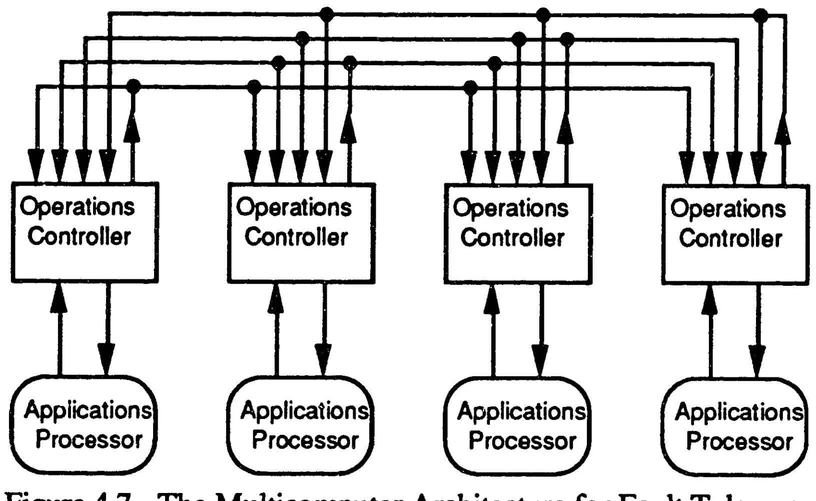


Figure 4.7 - The Multicomputer Architecture for Fault Tolerance

Seeing that there are four processors, the MAFT meets the participant number requirement for Byzantine resilience. All of the participants are connected to each other through unique broadcast busses. This meets the connectivity requirement. The two rounds of communication of the communication requirement take place as follows. In the first round, all of the the APs send their respective output data to the OC; this is equivalent to the commander giving his orders. The second round involves the relaying of this data to the other OCs, representing the intercommunication between the lieutenants. Finally, the MAFT has a synchronization scheme which synchronizes the participants to within a known skew (the synchronization requirement). This is done through the exchange of synchronization messages, which is also sent via the broadcast busses. Thus, the MAFT fulfills the requirements for One-Fault Byzantine resilience.

4.4 The Network Element Based Fault Tolerant Processor (NEFTP)

The Network Element Based Fault Tolerant Processor (NEFTP) [TAA88] is basically a scaled down version of the C1 FTPP (see Section 4.2) also developed at the Charles Stark Draper Laboratory, Inc. It consists of four Fault Containment Regions (FCRs), each containing a Processing Element (PE) and a Network Element (NE), as shown in Figure 4.8. As in the C1 FTPP, the PE is an off-the-shelf, VME compatible processor board. 20 MHz versions of the Motorola 68030 VME boards are used as PEs. The NE for the NEFTP is a reduced version of the NE for the C1 FTPP. Since there is only one PE per NE, the designs of the NE Global Controller and NE Scoreboard are extremely simplified. The C1 FTPP's NE occupies three VME boards as opposed to one VME board for NEFTP's NE. Fiber optic links are used instead of metallic links as the Inter-FCR Communication Links. This provides for much better electrical isolation between the FCRs and is less noisy. The NEFTP uses Class 1, Class 2, and SERP message exchanges similar to the C1 FTPP.

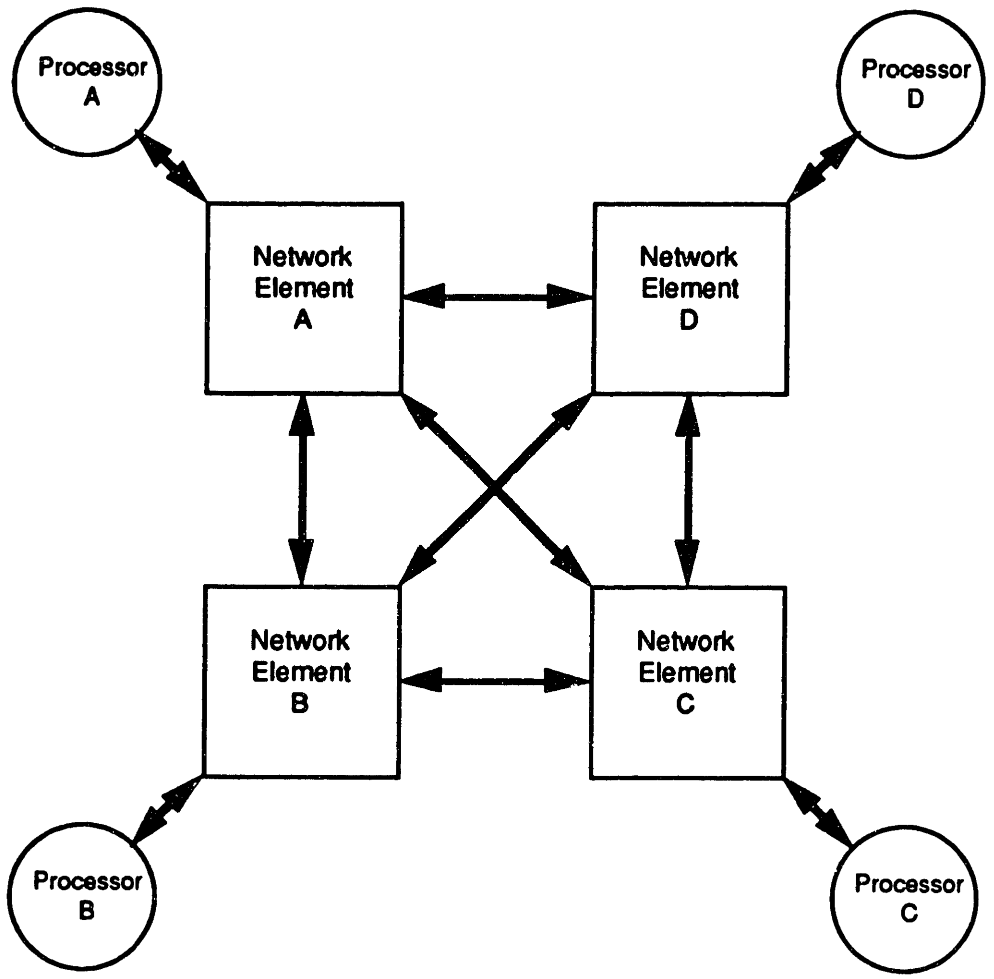


Figure 4.8 - The NEFTP Architecture

Each of the four FCRs in the NEFTP is equivalent to a participant in the Byzantine Generals Problem. This fulfills the participant number requirement for Byzantine resilience. Each NE is connected to the other three NEs via unique fiber optic links, meeting the connectivity requirement. A Class 2 exchange of data during the simplex mode of operation meets the communication requirement for Byzantine resilience since it is equivalent to two rounds of communication. The NEs are synchronized by the Fault Tolerant Clock mechanism, as in the C1 FTPP, fulfilling the synchronization requirement. In triplex or quadruplex modes, data is majority voted after one round of exchange, thus allowing one fault per exchange. The NEFTP meets all the requirements for being One-Byzantine resilient.

4.5 The Software Implemented Fault Tolerance Computer (SIFT)

The Software Implemented Fault Tolerance Computer (SIFT) [JG84, JHW78, RWB86] is a fault tolerant computer developed by SRI International for NASA to be used in a commercial air transport FBW. The SIFT consists of four to six processors with the processors being fully interconnected. The mechanism for synchronization and fault tolerance of the system is done through software, which leads to serious performance problems.

Having four to six fully interconnected processors, the SIFT fulfills the participant number and connectivity requirements for One-Byzantine resilience. Two rounds of communication take place as follows. First, each processor executes an identical software task (equivalent to the commander's orders). Secondly, each processor relays its output to the other processors for consensus. This meets the communication requirement. To synchronize the system, each processor broadcasts the value of its local clock to the other processors and adjusts its own clock to the other processors, meeting the synchronization requirement. The SIFT, therefore, is a One-Fault Byzantine resilient computer.

Chapter 5
The C3 Fault Tolerant Parallel Processor

The Cluster Three (C3) version of the Fault Tolerant Parallel Processor (FTPP) is under development at the Charles Stark Draper Laboratory. The C3 FTPP will be similar to the C1 FTPP discussed in Section 4.2, with several enhancements. It will contain up to eight processor elements (PEs) in each network element (NE) instead of the four PEs as in the C1 FTPP. It will also contain up to five NEs instead of four, for a total of forty PEs as opposed to sixteen in the C1 FTPP, which will greatly improve the throughput. The physical size of the NE will be reduced from three full size VME boards to one full size board. The reduced size improves on both the power dissipation and reliability of the C1 FTPP. The architecture for the C3 FTPP is diagrammed in Figure 5.1.

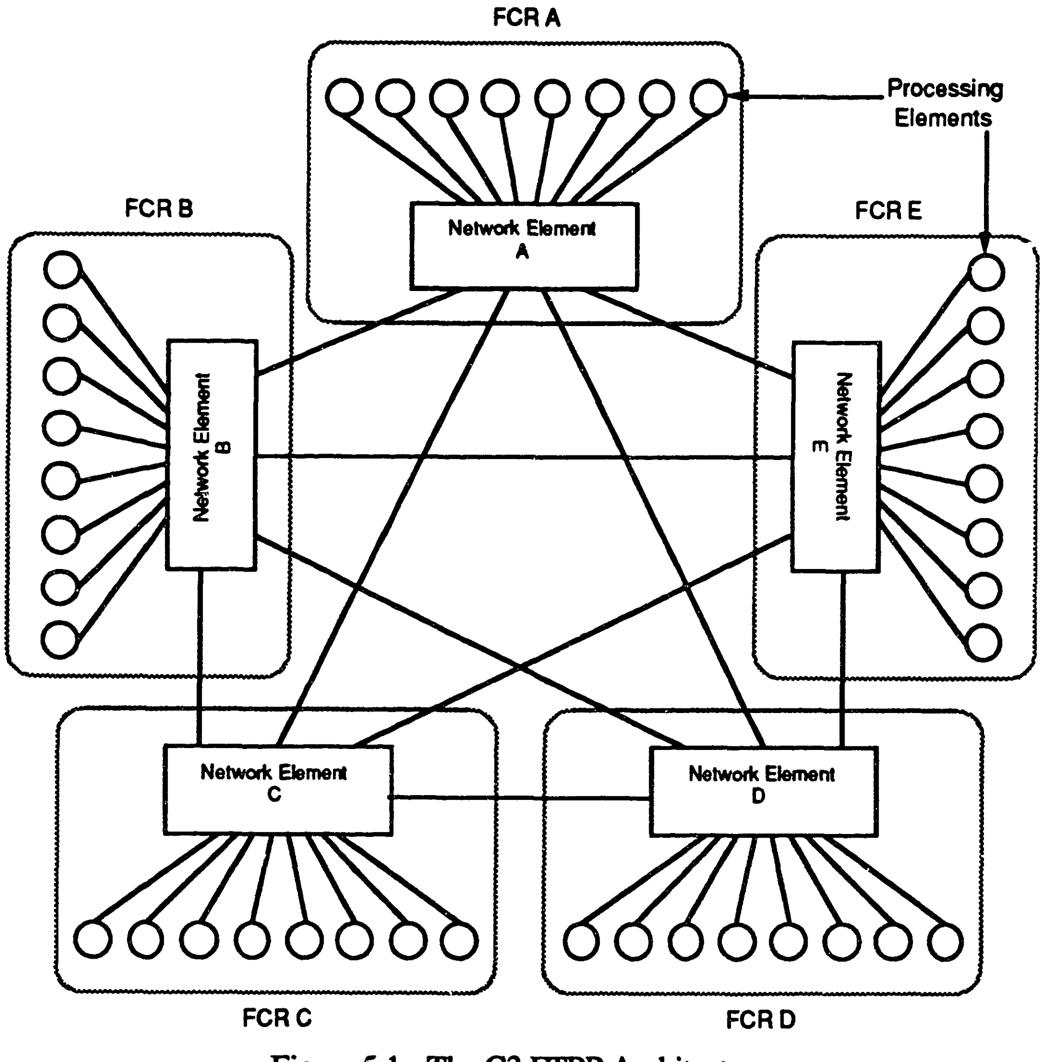


Figure 5.1 - The C3 FTPP Architecture

The five network elements are absolutely referenced as NE A, NE B, NE C, NE D, and NE E, as shown in Figure 5.1, or as 000, 001, 010, 011, and 100, respectively, when represented in binary. Relative addressing is also used when referencing a NE's relative position. Channel 0 (CH 0) references the local NE itself. CH 1 references the NE immediately to the right of the local NE as shown in Figure 5.1. Successive channels are assigned to each NE in counter-clockwise order, so CH 4 references the NE immediately to the left of the local NE. For example, when NE D represents CH 3 with respect to NE B.

5.1 C3 FTPP Hardware

The NE of the C3 FTPP consists of six major sections, the PE/NE Interface, the Scoreboard, the Global Controller, the Fault Tolerant Clock, the Data Paths/Voter, and the Inter-FCR Communication Links. The block diagram of the NE is shown in Figure 5.2. Fiber optic links are be used for the Inter-FCR Communication Links instead of metallic VME links to provide noise reduction and electrical isolation between NEs. The FTC FPGA designed in this thesis represents the NE Fault Tolerant Clock (FTC) and generates signals that control the Inter-FCR Communication Links. The Voter FPGA represents most of the NE Data Paths/Voter. The gray sections of Figure 5.2 represent sections that the FTC and Voter FPGAs control. Using FPGAs as opposed to standard TTL integrated circuits results in a reduction in noise due to fewer signal lines, lower power consumption, and a smaller physical NE size. Since there are twice as many PEs in each FCR and an additional NE in the C3 FTPP, the PE/NE Interface, NE Scoreboard, and NE Global Controller sections are also revised.

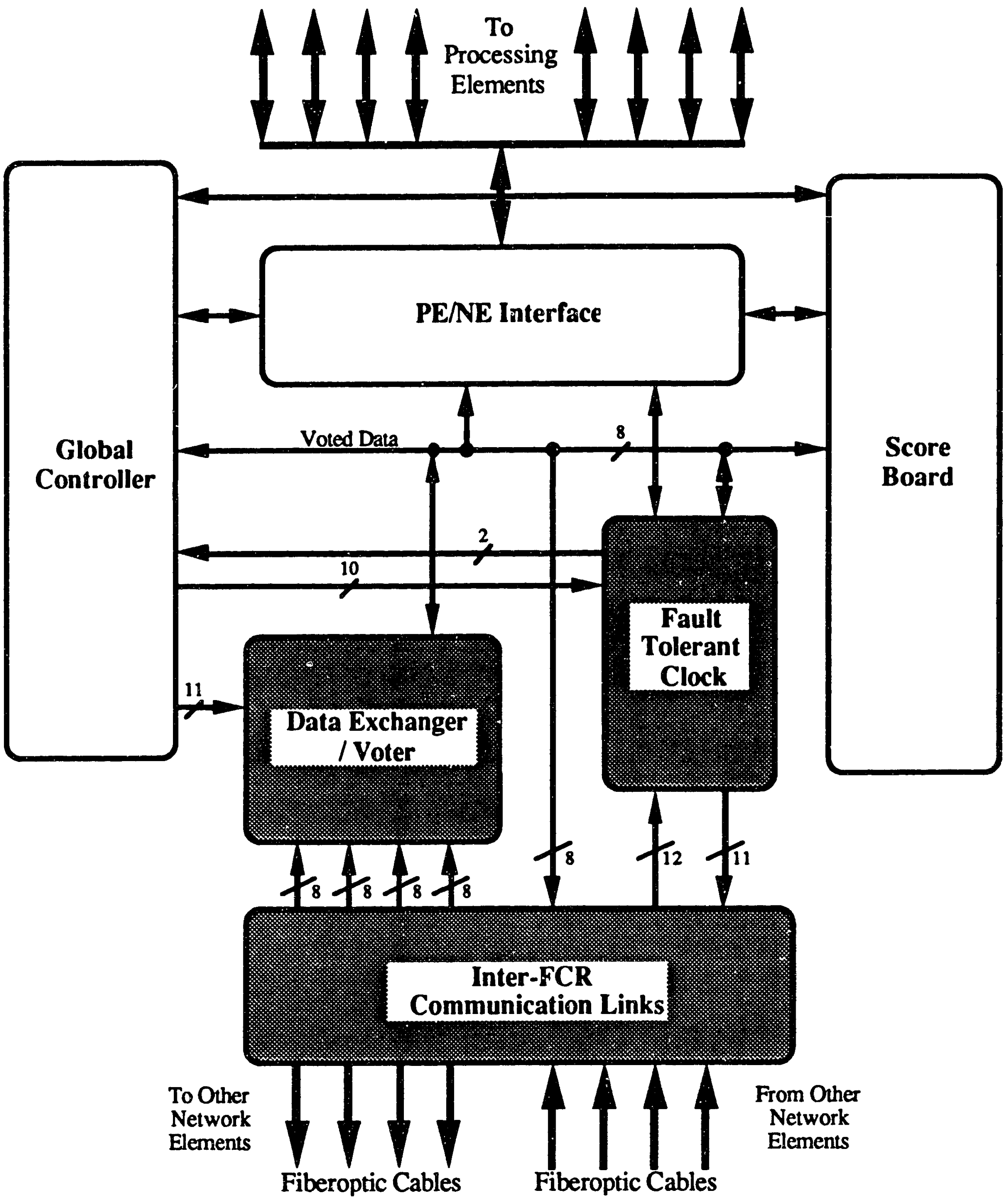


Figure 5.2 - Block Diagram of the C3 FTPP Network Element

The Inter-FCR Communication Links consists of one transmitting AMD TAXI chip, which transmits data from the voted data bus, and four receiving AMD TAXI chips, one for each incoming channel. The FTC FPGA synchronizes the local NE with respect to the other NEs by synchronizing itself to the message frames of the other NEs, recording errors when NEs are offsync. It also generates signals that control the Inter-FCR

Communication Links, recording transmission errors. The Voter FPGA majority votes maskable data from five IDT7202 FIFOs, one for each channel.

The FTC FPGA has an FTC section that handles the synchronization and recording of synchronization errors. This FTC section can be further divided into state machines, latches, and counters. The FTC FPGA has an Asynchronous Control section which controls the loading of the CH1, CH2, CH3, and CH4 FIFOs, which normally occur asynchronously with respect to the local NE's clock. Debug modes are supported, which allow data from an NE to be reflected to itself, so that an NE may be tested individually. A Synchronous Control section in the FTC FPGA controls the shifting out of data from the five FIFOs and shifts data into the CH0 FIFO. Transmission errors are recorded by the Link Error Accumulator in the FTC FPGA. The hardware details of the FTC FPGA are discussed in Chapter 7.

The Voter FPGA has a Voter section which majority votes five eight-bit wide maskable channels of data. It generates any errors that result. The Mask Register section of the Voter FPGA generates the various masks that are used by the Voter. The Syndrome Accumulator records errors that occur from the Voter section. The hardware details of the Voter FPGA are discussed in Chapter 8.

The C3 FTPP hardware will support any microprocessor that is VME compatible. Currently, C3 FTPP software will support the 68030 microprocessor from Motorola. However, the 68040 microprocessor from Motorola, the R3000 RISC processor, and the 80386 and 80960 microprocessors from Intel, may be supported at a later date. The processors of the C3 FTPP are arranged as virtual processors, each consisting of one, three, or four physical processors to provide different levels of fault tolerance and throughput. A virtual processor consisting of one, three, or four physical processors is called a simplex, triplex, or quadruplex virtual processor, respectively. The duplex virtual processor is not supported due to the fact that errors can only be detected and not corrected. Note that there is no quintuplex virtual processor even though five network elements do exist. Each of the processors in the virtual processing group executes the same code. The members of a virtual processor are called a virtual group. Each of the redundant processors in a virtual processing group reside in a separate Fault Containment Regions (FCRs) to eliminate single point failures. Communication takes place between the processors in a virtual group by means of the NE. PEs in a virtual group can communicate with other PEs in other virtual groups by means of the NE. Each PE receives identical messages, which are delivered synchronously. By this means, a virtual group can synchronize itself by sending itself a message and awaiting its return. The NEs also synchronize themselves upon power up by means of the global controller. The details of the synchronization are discussed in the Fault Tolerant Clock section (Section 7.3).

The C3 FTPP supports dynamic reconfiguration, which allows real-time reconfiguration of the physical processors in a virtual group. This reconfiguration is aided by the NE Scoreboard, which contains a configuration table (CT) keeping track of the current configuration of the C3 FTPP. One virtual group, called the reconfiguration authority, is responsible for the updating the configuration table. This group must be a triplex or quadruplex group, assuring one-fault Byzantine resilience.

The C3 FTPP is One-Fault Byzantine resilient. Having five NEs fulfills the participant number requirement. Each NE is inter-connected via fiber optic links, meeting the connectivity requirement. A Class 2 exchange, discussed in the next section, meets the two round communication requirement. The NEs are synchronized by the Fault Tolerant Clock mechanism, fulfilling the synchronization requirement.

5.2 C3 FTPP Exchange Protocol

To maintain synchronization and consistency, the processors exchange messages. Each NE tells the other NEs whether or not its processor needs to exchange information. If all non-faulty processors request the exchange, then the exchange is performed. This cycle is diagrammed in Figure 5.3.

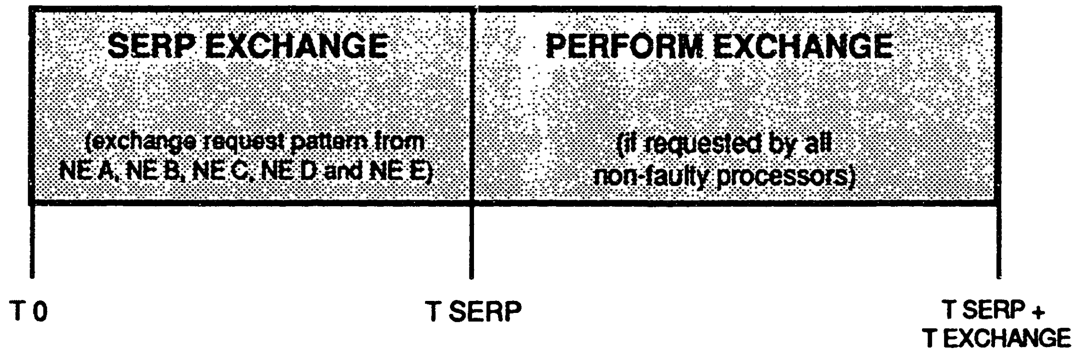
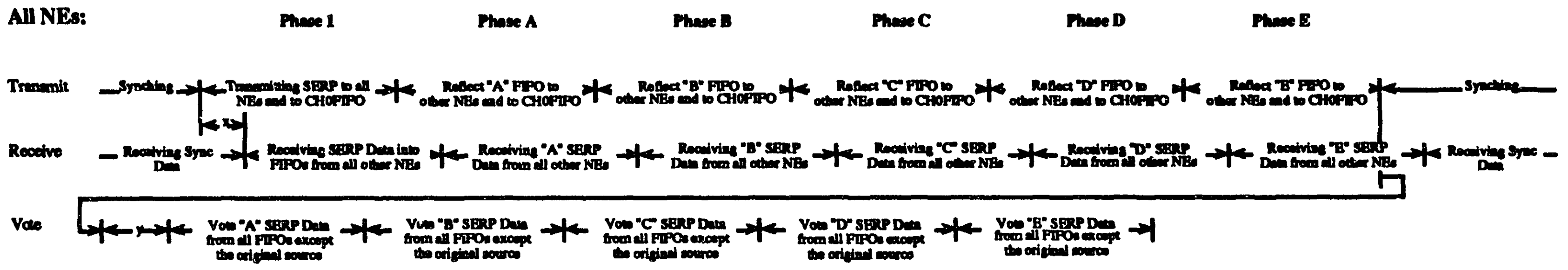

SERP EXCHANGE
(exchange request pattern from NE A, NE B, NE C, NE D and NE E)
PERFORM EXCHANGE
(if requested by all non-faulty processors)
T 0
T SERP
T SERP + T EXCHANGE

Figure 5.3 - Basic C3 FTPP Network Element Cycle

The data sent by a NE requesting an exchange of information is called an exchange request pattern. Five of these exchanges occur during the first half of this cycle. The series of these five exchanges is called a system exchange request pattern (SERP) exchange. The timing of the SERP Exchange is diagrammed in Figure 5.4.


All NEs:
Phase 1
Phase A
Phase B
Phase C
Phase D
Phase E
Transmit
Synching
Transmitting SERP to all NEs and to CH0FIFO
Reflect "A" FIFO to other NEs and to CH0FIFO
Reflect "B" FIFO to other NEs and to CH0FIFO
Reflect "C" FIFO to other NEs and to CH0FIFO
Reflect "D" FIFO to other NEs and to CH0FIFO
Reflect "E" FIFO to other NEs and to CH0FIFO
Synching
x
Receive
Receiving Sync Data
Receiving SERP Data into FIFOs from all other NEs
Receiving "A" SERP Data from all other NEs
Receiving "B" SERP Data from all other NEs
Receiving "C" SERP Data from all other NEs
Receiving "D" SERP Data from all other NEs
Receiving "E" SERP Data from all other NEs
Receiving Sync Data
Vote
y
Vote "A" SERP Data from all FIFOs except the original source
Vote "B" SERP Data from all FIFOs except the original source
Vote "C" SERP Data from all FIFOs except the original source
Vote "D" SERP Data from all FIFOs except the original source
Vote "E" SERP Data from all FIFOs except the original source

Delay x: Propagation delay between NEs ± maximum skew between NEs.
Delay y: The time NE waits from start of data transmission to start of vote (Max NE skew + FIFO write to read time.)

Figure 5.4 - C3 FTPP SERP Exchange Timing

The first phase of a SERP Exchange involves each NE transmitting its SERP data to all the other NEs and to itself (CH0FIFO). This is diagrammed in Figure 5.5, which shows the data contained in each NEs FIFOs.

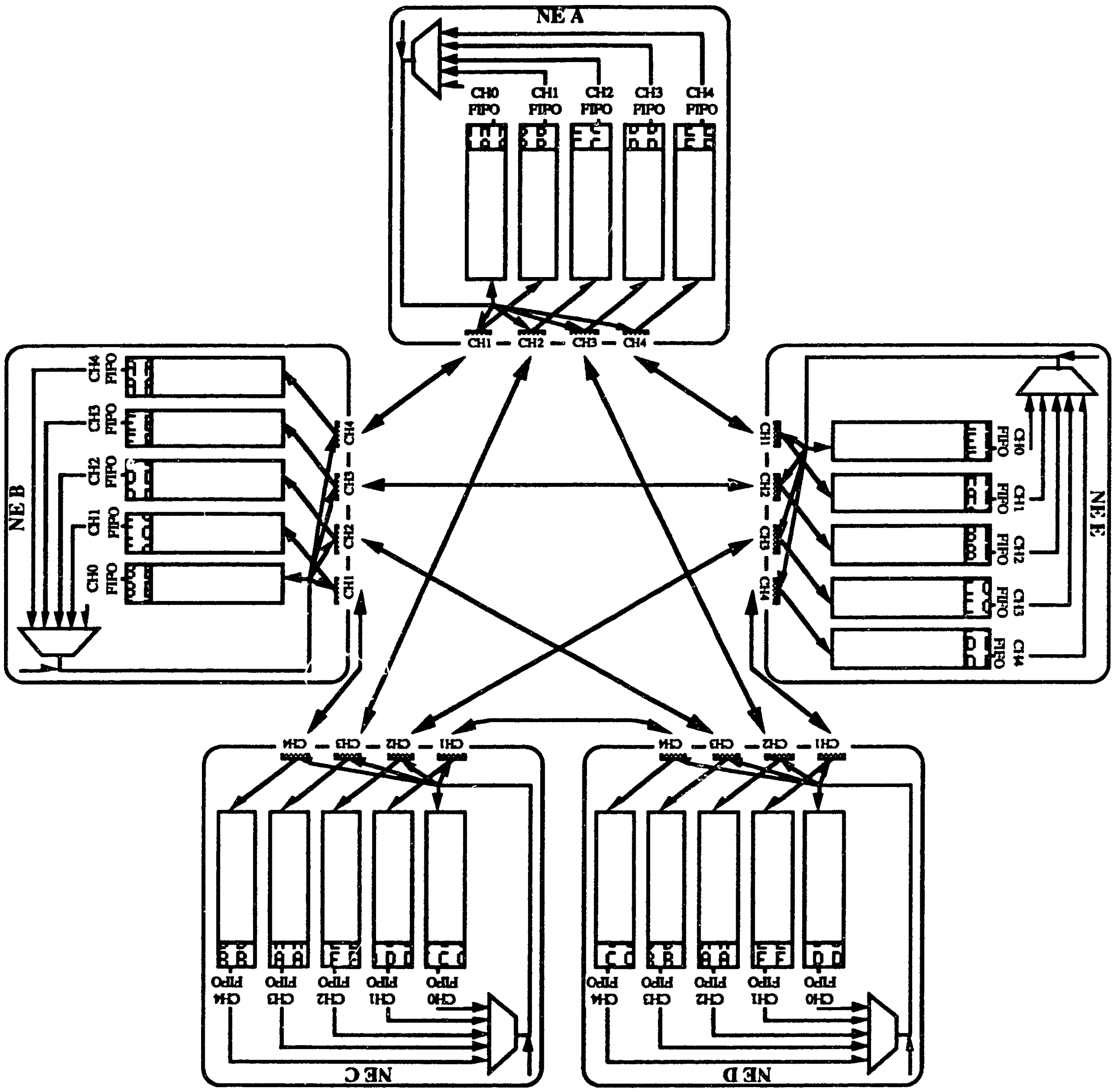


Figure 5.5 - C3 FTPP SERP Phase One

Phase A of a SERP Exchange involves each NE reflecting the NE A FIFO's data to all the other NEs and to itself. This is diagrammed in Figure 5.6.

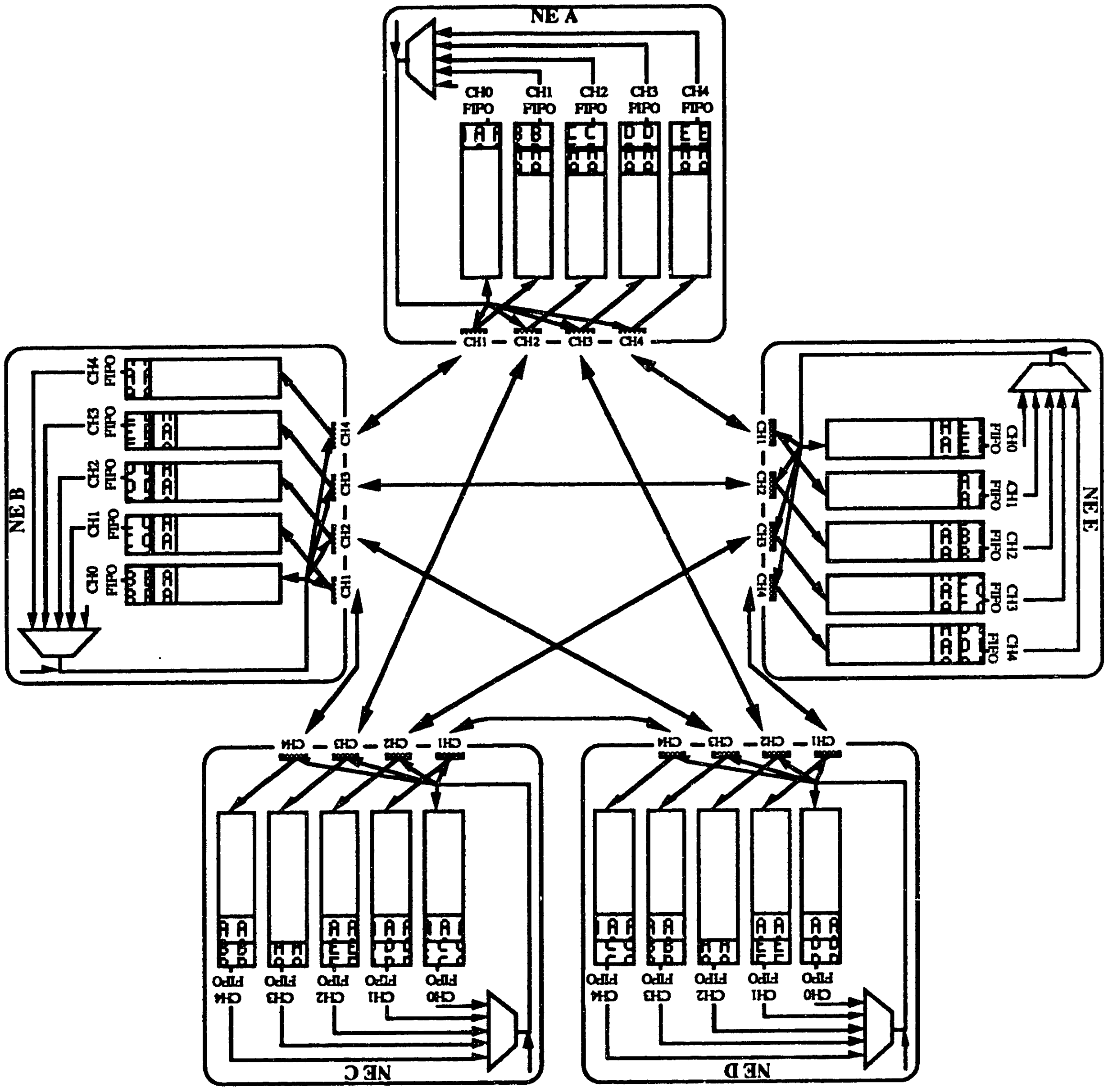


Figure 5.6 - C3 FTPP SERP Phase A

Phase B of a SERP Exchange involves each NE reflecting the NE B FIFO's data to all the other NEs and to itself. This is diagrammed in Figure 5.7.

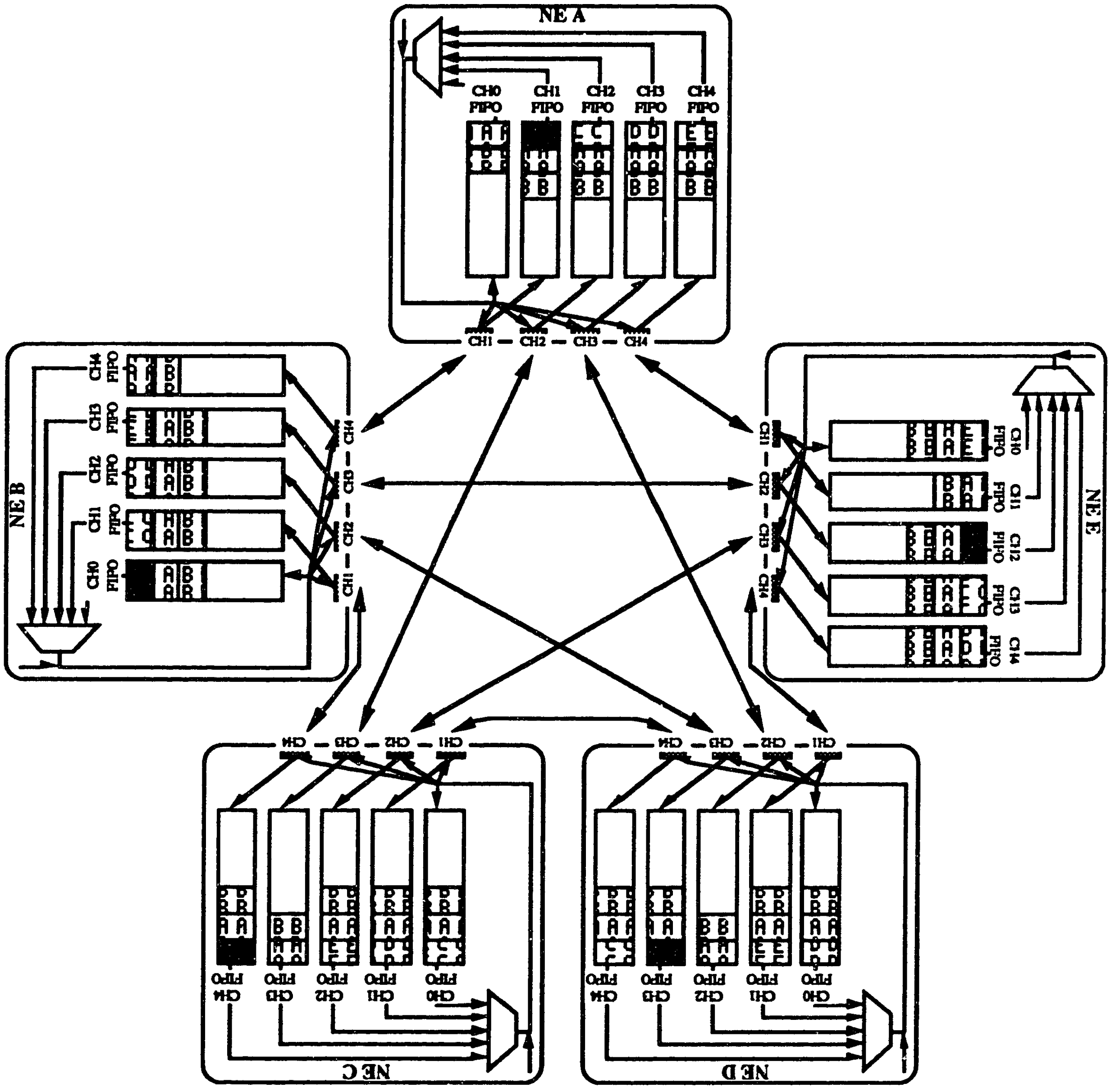


Figure 5.7 - C3 FTPP SERP Phase B

Phase C of a SERP Exchange involves each NE reflecting the NE C FIFO's data to all the other NEs and to itself. This is diagrammed in Figure 5.8.

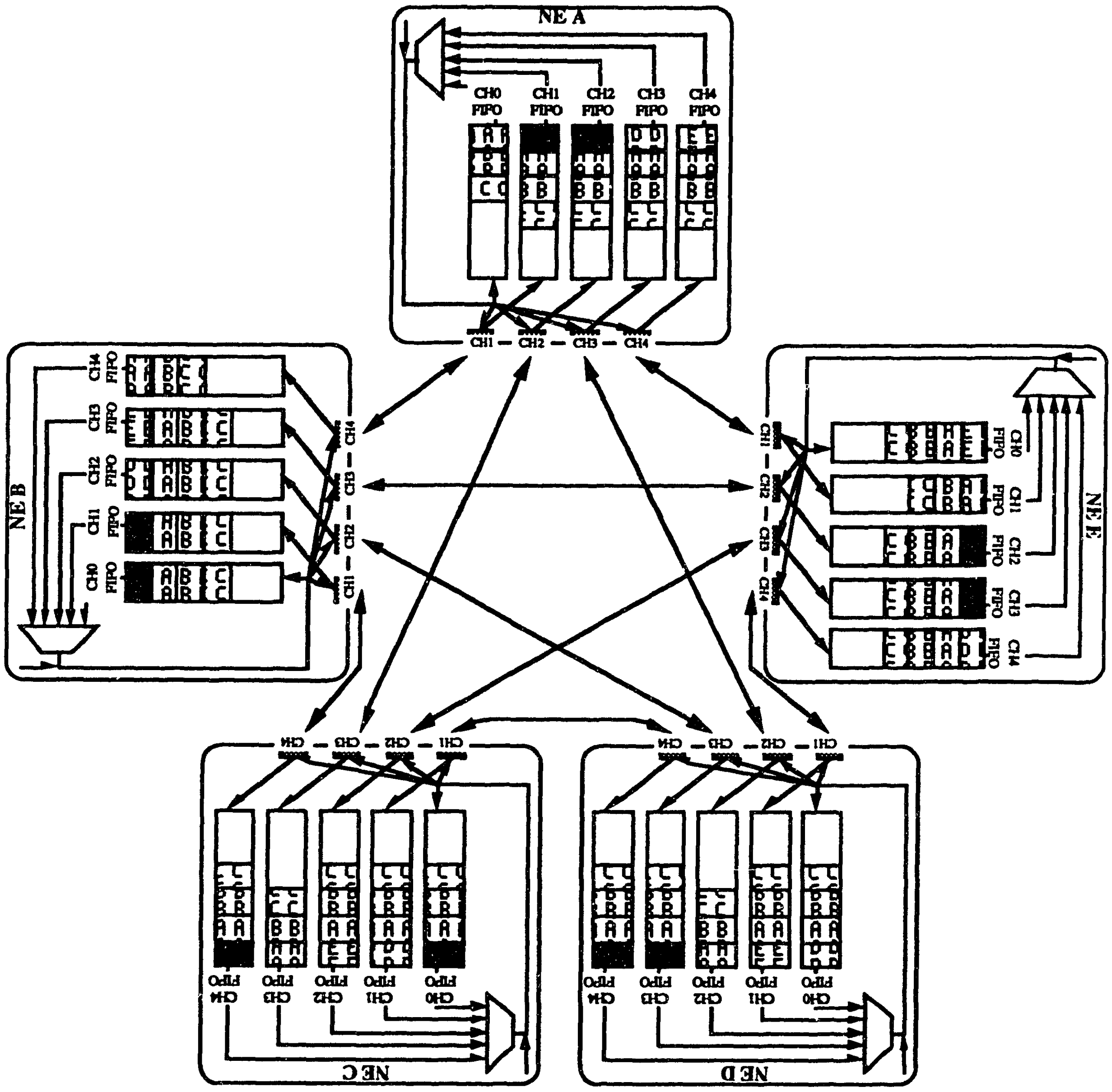


Figure 5.8 - C3 FTPP SERP Phase C

Phase D of a SERP Exchange involves each NE reflecting the NE D FIFO's data to all the other NEs and to itself. This is diagrammed in Figure 5.9.

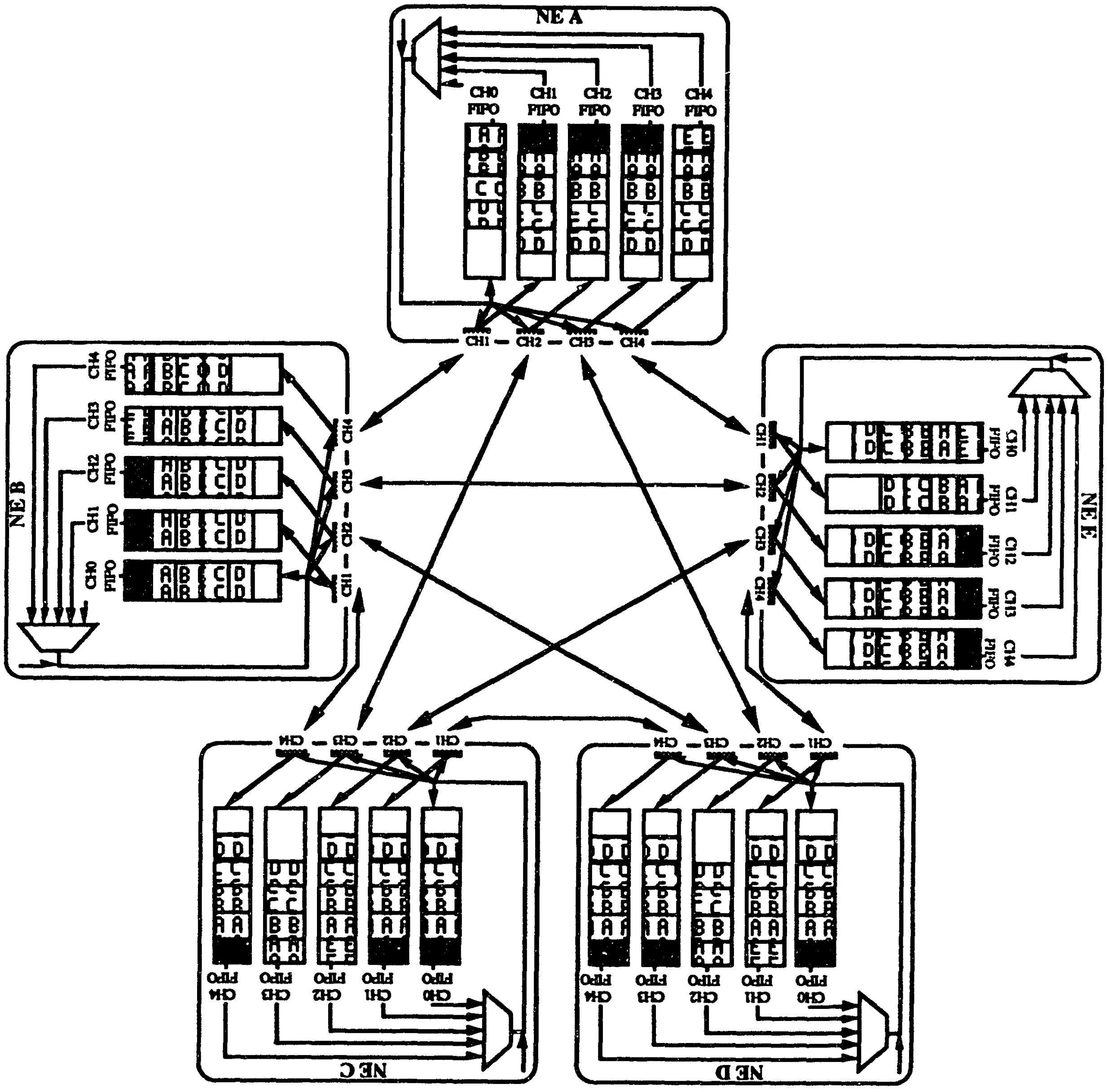


Figure 5.9 - C3 FTPP SERP Phase D

Phase E of a SERP Exchange involves each NE reflecting the NE E FIFO's data to all the other NEs and to itself. This is diagrammed in Figure 5.10.

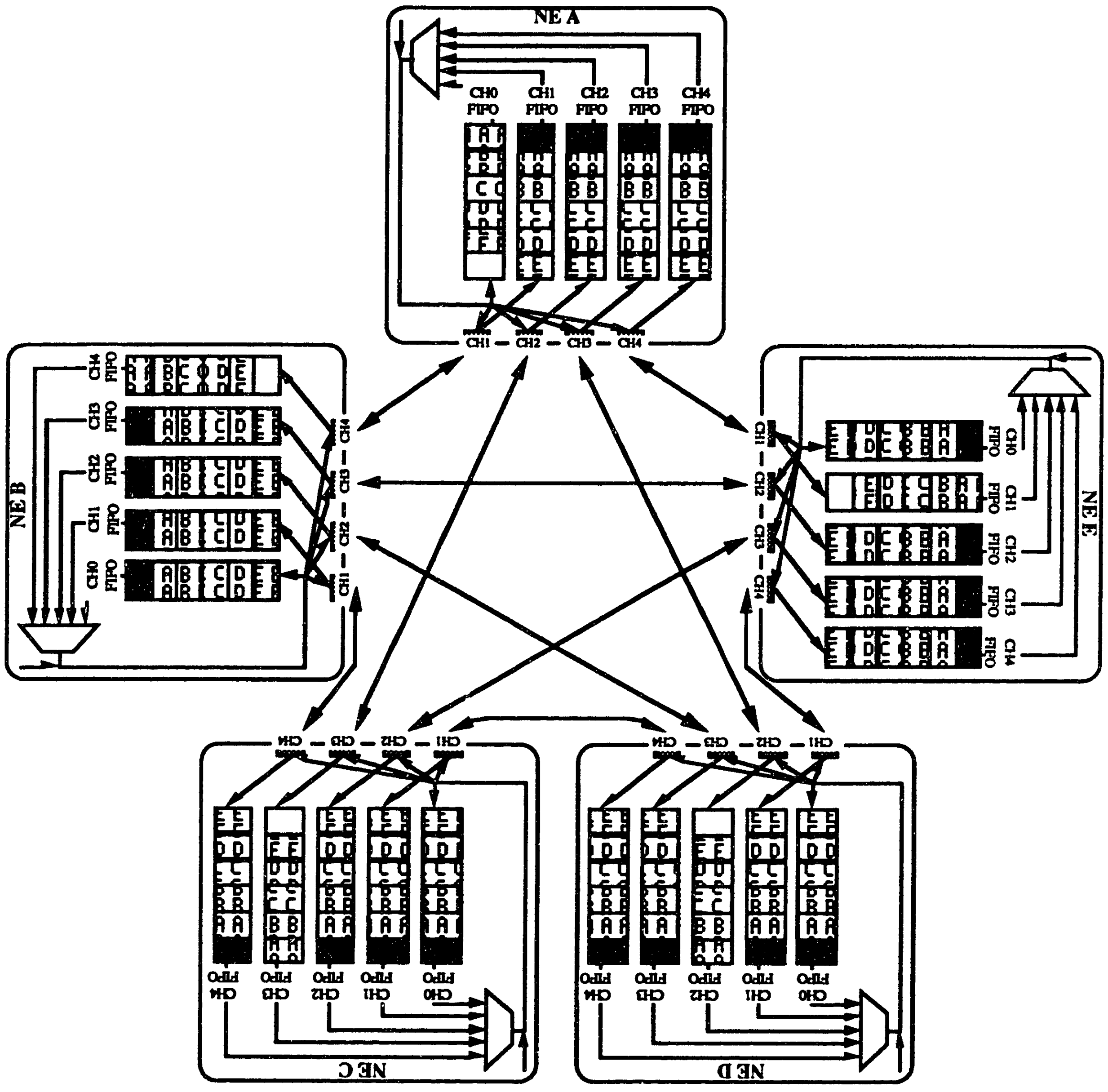


Figure 5.10 - C3 FTPP SERP Phase E

Once it is determined through a SERP that an exchange is to take place, there are two types of exchanges that can take place. The first type is a Class 1 exchange which is sourced by the members of a triplex, or quadruplex virtual processor. During this type of exchange, each member of the virtual processor delivers its message to its respective NE, which then relays the message to the other four NEs. Each NE not hosting a member of the virtual processor delivers an empty message to the other NEs. Note that all NEs transmit messages, so each NE transmits one message and receives four messages. Messages are transmitted and received by the Inter-FCR Communication Links. The information received by each NE is then majority voted using the Voter FPGA. Disagreements in the vote are relayed to the processing elements so that Fault Detection Isolation and Reconfiguration (FDIR) software can determine if an element is faulty and reconfigure the system as necessary. The Class 1 message exchange is diagramed in Figure 5.11. The PEs labeled with a T1 represent members of a triplex virtual processor.

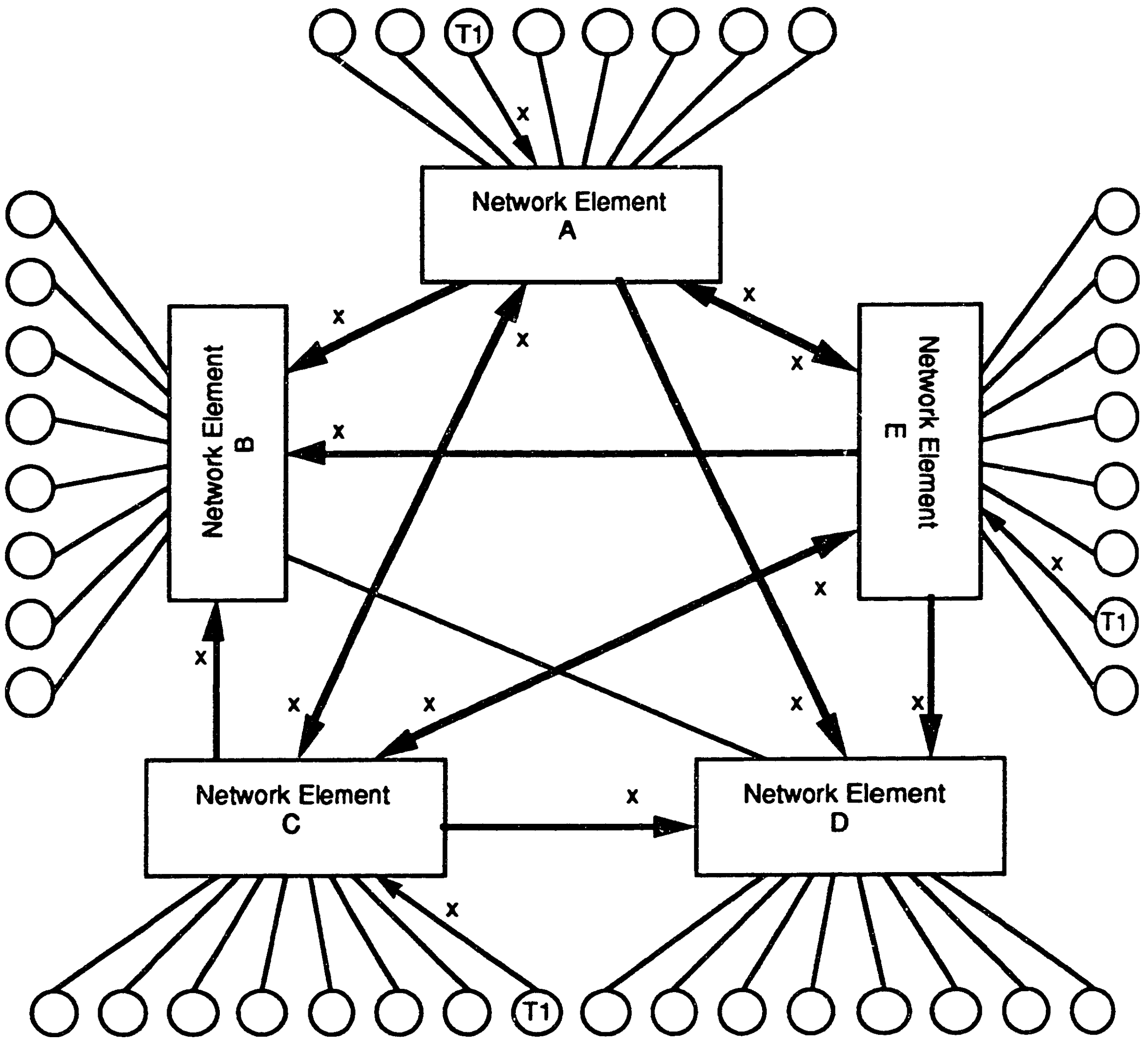
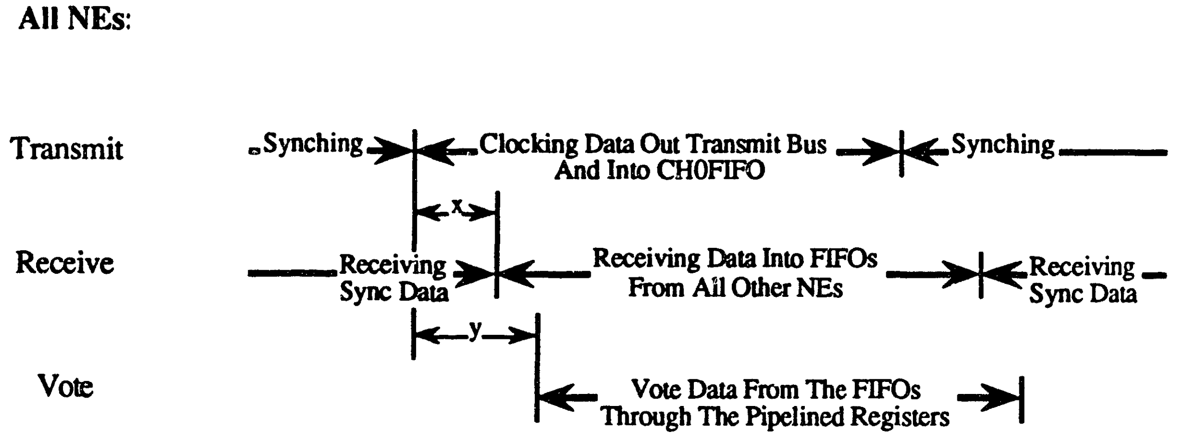


Figure 5.11 - C3 FTPP Class 1 Message Exchange

The Class 1 message exchange timing is diagrammed in Figure 5.12.


All NEs:
Transmit
Receive
Vote
Synching
Clocking Data Out Transmit Bus And Into CH0FIFO
Synching
x
Receiving Sync Data
Receiving Data Into FIFOs From All Other NEs
Receiving Sync Data
y
Vote Data From The FIFOs Through The Pipelined Registers

Delay x: Propagation delay between NEs ± maximum skew between NEs.

Delay y: The time NE waits from start of data transmission to start of vote (Max NE skew + FIFO write to read time.)

Figure 5.12 - C3 FTPP Class 1 Exchange Timing

For a simplex virtual processor, a Class 2 message exchange is used. A Class 2 exchange is a two-round exchange. In the first round, a member of the FMG single-sources a message to its hosting NE, which then relays this message to the other four NEs. In the second round, each NE reflects the message they received to the other four NEs. Thus, each NE has five copies of the message which is then majority voted. The Class 2 message exchange from NE A is diagramed in Figures 5.13 and 5.14. The PE labeled S1 represents a simplex virtual processor.

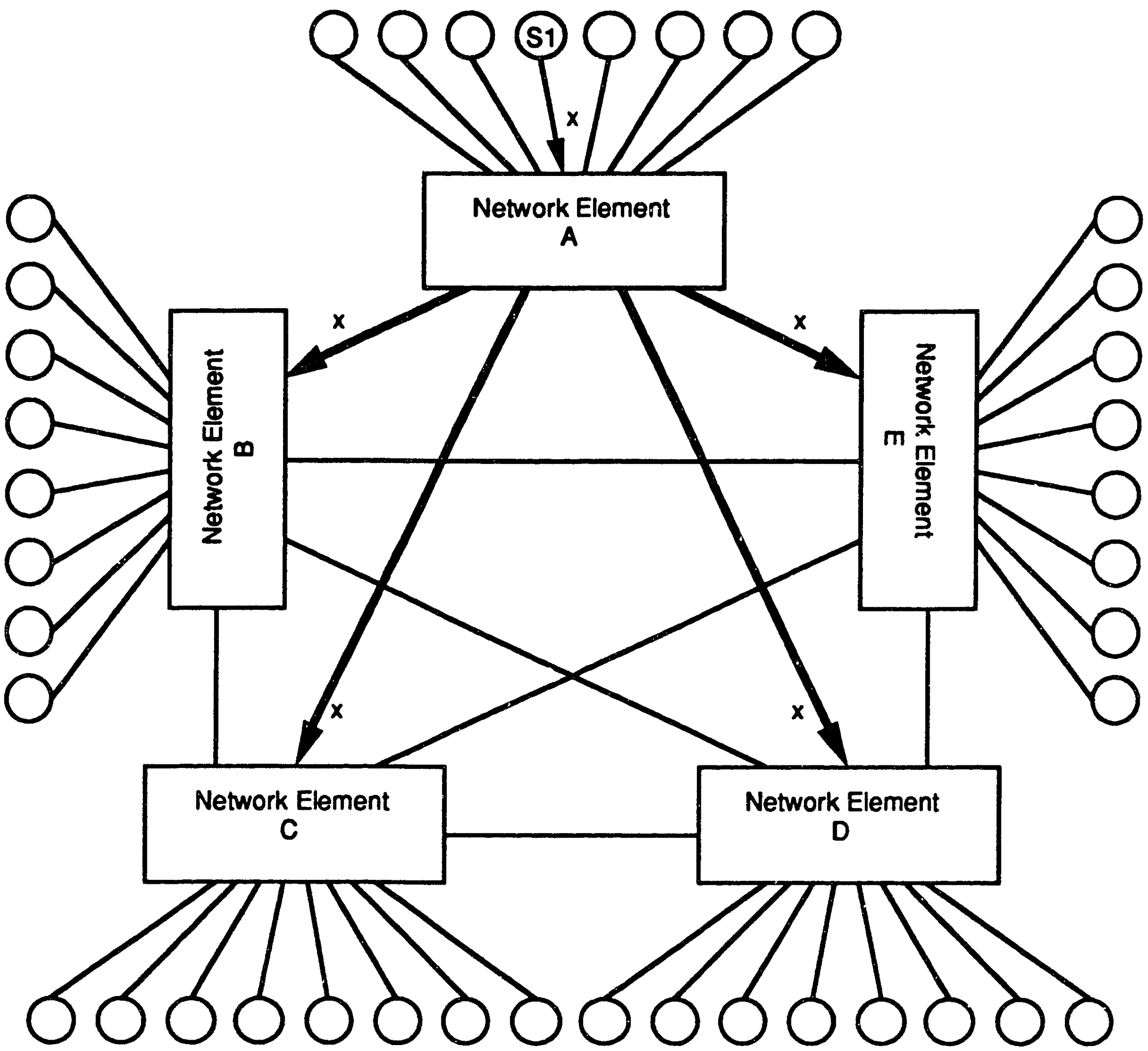


Figure 5.13 - C3 FTPP Class 2 Message Exchange (Round 1)

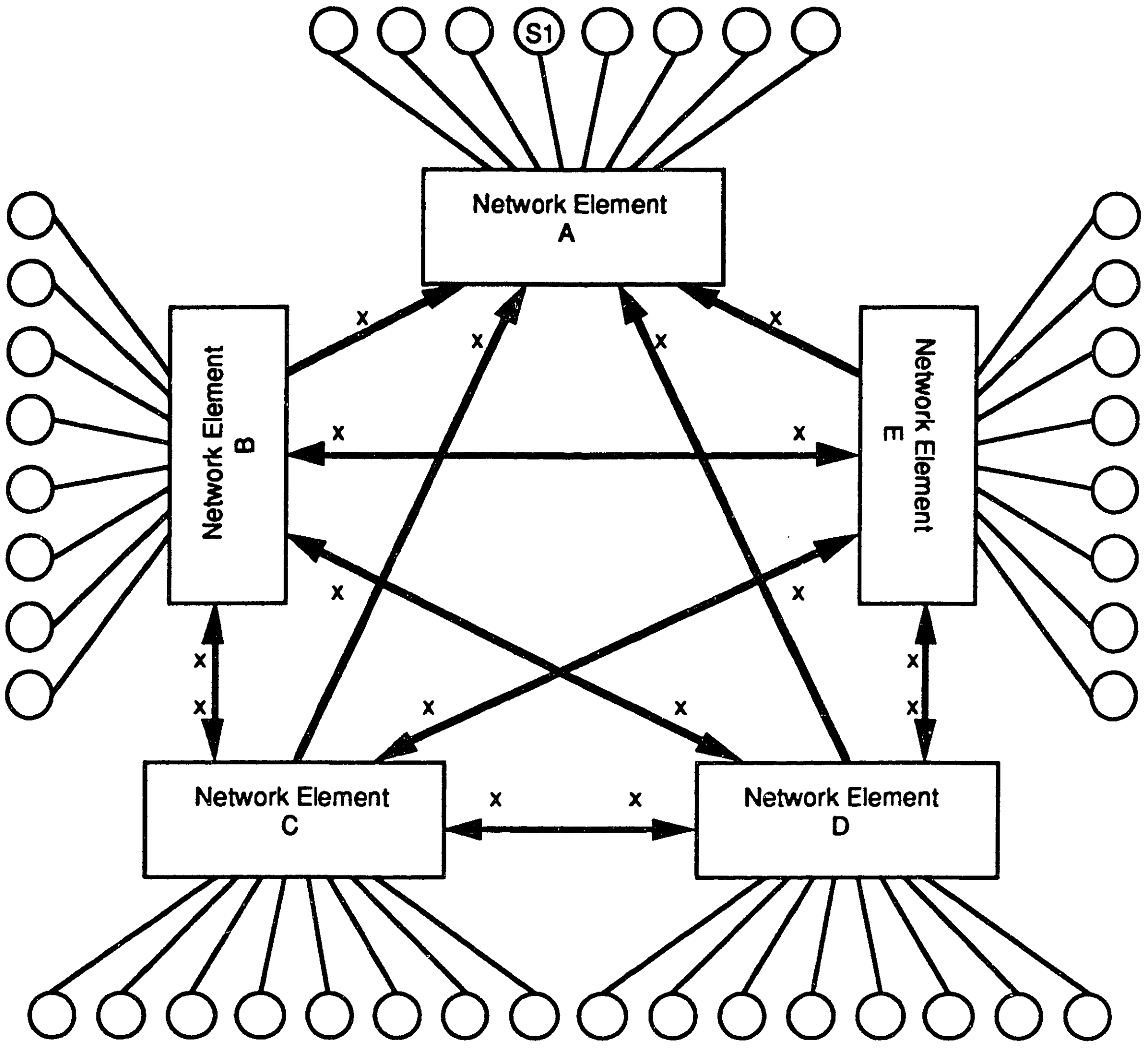


Figure 5.14 - C3 FTPP Class 2 Message Exchange (Round 2)

The Class 2 message exchange from NE A timing is diagrammed in Figure 5.15.

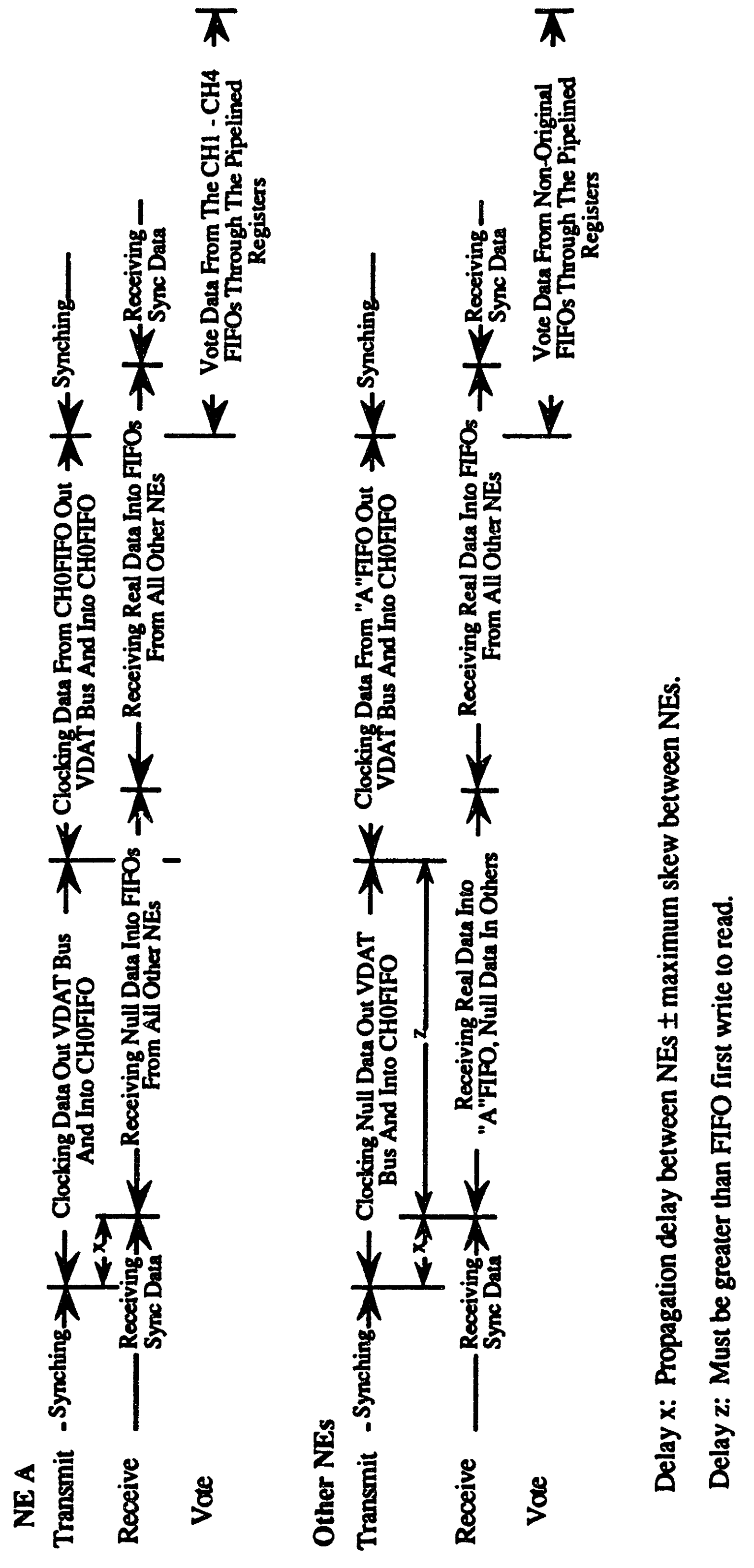

NE A
Transmit
Synching
Clocking Data Out VDAT Bus And Into CH0FIFO
Clocking Data From CH0FIFO Out VDAT Bus And Into CH0FIFO
Synching
Receive
Receiving Sync Data
x
Receiving Null Data Into FIFOs From All Other NEs
Receiving Real Data Into FIFOs From All Other NEs
Receiving Sync Data
Vote
Vote Data From The CH1 - CH4 FIFOs Through The Pipelined Registers
Other NEs
Transmit
Synching
Clocking Null Data Out VDAT Bus And Into CH0FIFO
Clocking Data From "A"FIFO Out VDAT Bus And Into CH0FIFO
Synching
Receive
Receiving Sync Data
x
z
Receiving Real Data Into "A"FIFO, Null Data In Others
Receiving Real Data Into FIFOs From All Other NEs
Receiving Sync Data
Vote
Vote Data From Non-Original FIFOs Through The Pipelined Registers
Delay x: Propagation delay between NEs ± maximum skew between NEs.
Delay z: Must be greater than FIFO first write to read.

Figure 5.15 - C3 FTPP Class 2 Exchange From NE A Timing

5.3 The Data Path Chip Set

The C3 FTPP Data Path Chip Set, which includes the FTC FPGA and the Voter FPGA, performs all the functions of the Fault Tolerant Clock section, generates signals that control the Inter-FCR Communication Links, and performs most of the functions of the Data Path section of the NE. The only part of the Data Path section not included in the Data Path Chip Set are five IDT7202 2K by 8 FIFOs, since its inclusion would overload the pin and gate capacity of the FPGA. The Inter-FCR Communications Links consists of four receiving fiber optic data links (FODLs) and one transmitting FODL that drives a one to four splitter, which sends data to each of the other four NEs. Each of the FODLs are driven by an AMD TAXI chip which converts eight bits of parallel data into fiber optic serial data, or vice versa. The block diagram of the Data Path Chip Set is shown in Figure 5.16.

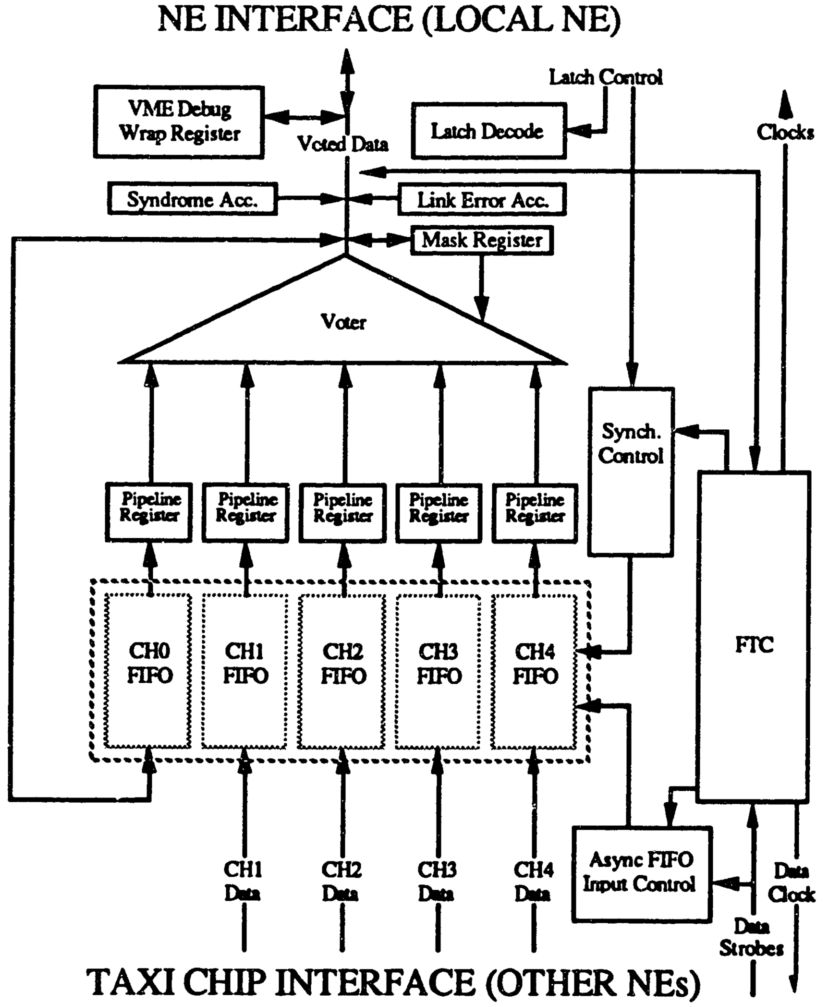


Figure 5.16 - Block Diagram of C3 FTPP Data Path Chip Set

The Data Path Chip Set allows the physical size of the C3 FTPP to be more compact, to be more reliable, and to have a lower power dissipation as opposed to a design done with PALs and TTL chips. If the equivalent of the Data Path Chip Set were to be designed using PALs and TTL chips, the design could be implemented using thirteen 24-pin PALs, seventeen 20-pin PALs, five 74AS374 octal latches, two 74LS74 latches, and four 74LS163 counters. For a size comparison, see Figures 5.17 and 5.18.

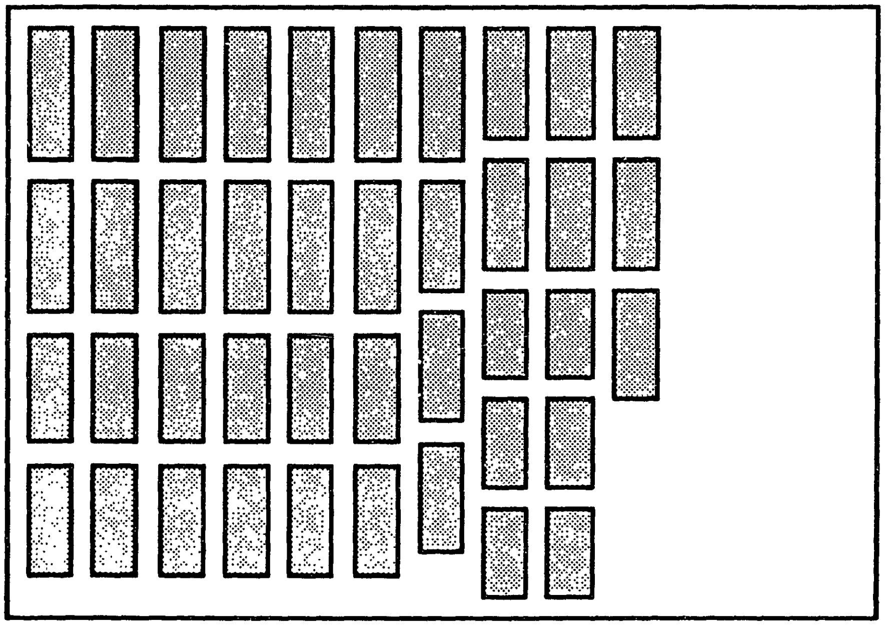


Figure 5.17 - Equivalent of Data Path Chip Set Using PALs and TTL logic

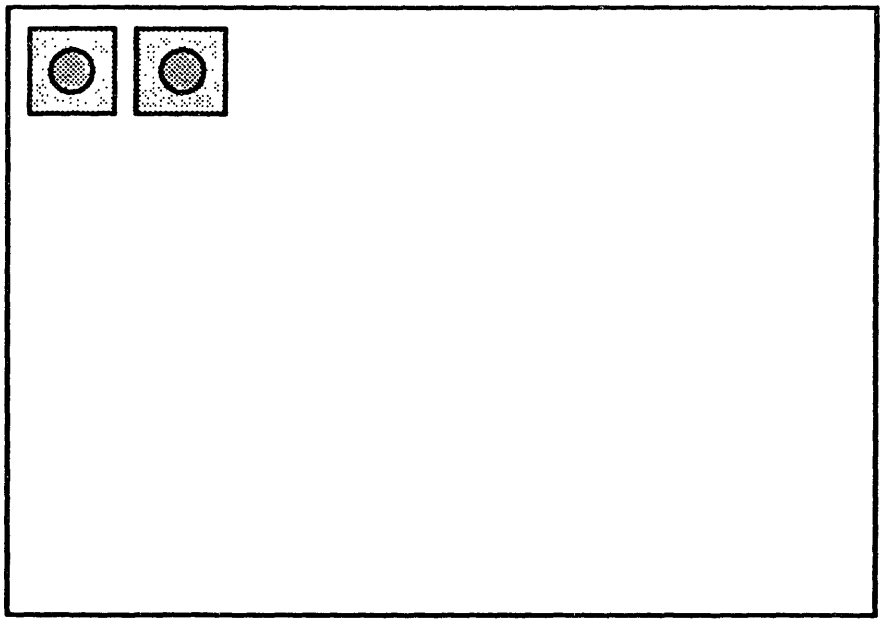


Figure 5.18 - Data Path Chip Set Using Altera FPGAs

The PAL and TTL implementation of the Data Path Chip Set would have a total power dissipation of approximately thirty-three watts, as opposed to approximately three watts using Altera FPGAs.

The failure rate for the FPGA implementation would also be reduced. To get an estimate, we can calculate the estimated reliability by using formulas given in the 1986 Military Handbook for Reliability Prediction of Electronic Equipment. The formulas used in this handbook use the chip gate counts, pin counts, and power dissipations to calculate an estimate of reliability. The PAL and TTL implementation of the Data Path Chip Set was calculated to have a reliability of 37.5 failures per one-million hours. The Altera FPGA implementation had a calculated reliability of 23.4 failures per one-million hours for our two chips. [MIL86] Thus, we can estimate a failure reduction of over 60% by using Altera FPGAs for the Data Path Chip Set design.

The MAX EPM5128 technology is based on EPROM memory technology, which has proven to be highly reliable. Intel has done a reliability study on their EPROM memories. The results seem to shown that memory size does not have a large effect on failure rate. For various EPROM memory sizes ranging from 4K x 8 to 128K x 8 memories, Intel has reported failure rates on the order of 0.01 per 1,000 hours. [INT86] Another approach to estimating the failure rate for the FPGAs is to estimate the failure rate of the non-EPROM parts and add the failure rate of the EPROM memory as reported by Intel. Using the same Military Handbook formulas as before, the reliability of our two Altera FPGAs not including the EPROM memory (vastly decreasing the gate count in our formula) is 6.1 failures per one-million hours. [MIL86] Adding Intel's failure rate of 10 failures per one-million hours, which itself includes pin output and other inherent failures, yields a combined failure rate of 16 failures per one-million hours, which is less than half the failure rate of our PAL and TTL implementation. Note that either approach generates only a component failure rate. It does not take into account interconnect failures. The PAL and TTL implementation of the Data Path Chip Set would have several times more interconnects than the FPGA implementation. Thus, the FPGA implementation of the Data Path Chip Set has a reliability enhancement of nearly three times tha. of the PAL and TTL implementation.

The AT & T Reliability manual estimates the lifetime reliability to be 25 failures per device for the octal latches, 10 per device for the D flip-flops, 25 per device of the counters, and 40 per device for the CMOS PALs. [DJK90] This yields a total of 1445 lifetime failures for our PAL and TTL implementation. The AT & T Reliability manual estimates the lifetime reliability of CMOS EPROM memories to be 100 failures per device and 50 failures per device for our equivalent CMOS equivalent device. [DJK90] This yields a total of 300 lifetime failures for two Altera FPGAs, which is a reduction of nearly five times the failures.

Although we get different results for the reliability figures using different methods, CMOS EPROM technology has proven to be very reliable and can expect a large increase in reliability. It is the author's opinion that the FPGA implementation would yield a reliability increase of at least three times that of the PAL and TTL implementation.

The FTC FPGA can be divided into the Asynchronous Control, Debug Wrap Register, Fault Tolerant Clock, Latch Decode, Link Error Accumulator, and Synchronous Control sections. The Asynchronous Control section controls the loading of the receiving FIFOs, which occurs asynchronously with respect to the local NE's clock. The Debug Wrap Register wraps a byte of data from the voted data bus onto a register for future access. The Fault Tolerant Clock (FTC) synchronizes the local NE with respect to the other NEs by synchronizing itself with respect to the other NEs' message frames. It also generates its own message frame. The Latch Decode section is a decoder that generates signals for use throughout the FTC FPGA. The Link Error Accumulator keeps track of errors that occur when a NE's clock is offsync with the other NEs' clocks. The Synchronous Control shifts data out of the FIFOs and shifts data into the CH0 FIFO, which occurs synchronously to the local NE's clock.

The Voter FPGA can be divided into the Latch Decode, Mask Register, Syndrome Accumulator, and Voter sections. The Latch Decode section is a decoder that generates signals for use throughout the Voter FPGA. The Mask Register generates masks that are used by the Voter section. The Syndrome Accumulator records errors that occur during the voting process. The Voter is a five-way maskable, byte-wide, bit-for-bit majority voter. Voting errors are also generated by the Voter.

Chapter 6
C3 FTPP Data Path Chip Set Hardware

When deciding on the technology to use for construction of the C3 FTPP NE Fault Tolerant Clock and NE Data Paths sections, many technologies could have been chosen, such as resistor-transistor logic (RTL), diode-transistor logic (DTL), transistor-transistor logic (TTL), emitter-coupled logic (ECL), integrated injection logic (I^2L), negative metal oxide silicon (NMOS), and complementary metal oxide silicon (CMOS). RTL and DTL technologies are obsolete. TTL and ECL are used mainly in small-scale integration (SSI) and medium-scale integration (MSI). The C1 FTPP was designed using standard TTL technology, and as a result, the NE occupied three full size VME boards. To reduce size and power dissipation in the C3 FTPP, the circuitry area must be reduced. This narrows our technology choices to I^2L, NMOS, and CMOS. Since much of the NE will still use standard TTL or TTL-compatible technology, CMOS technology was chosen for the Data Path Chip Set since interfacing between CMOS and TTL is much simpler than interfacing between I^2L and TTL or between NMOS and TTL.

This chapter discusses some of the technologies used for designing and fabricating CMOS VLSI integrated circuits.

6.1 Types of Design

There are several technologies available for developing a CMOS VLSI design. In general, VLSI design technologies can be placed into two categories, full custom or semi custom. Full custom, also known as handcrafted mask layout, refers to layout of the functional subsystems at the mask level [KE85]. This means that everything in the ASIC is laid out, down to the placement of transistors and their connections. This is done for speed and density purposes, but tends to have a very long design time and is very expensive. Semi custom design includes two major technologies, Standard Cell and Gate Array. Standard Cell technology involves designing the ASIC with predefined logic/circuit cells such as nand gates, nor gates, latches, etc. This reduces the design time greatly, has good speed and density characteristics, but is also very expensive. Gate Array technology involves designing the ASIC with prefabricated transistor gate arrays, so that only connections between gates are laid out on the chip. The design time is also fast, and manufacture is the least expensive of the three technologies. However, ASICs designed in this manner are not as dense and fast as those designed by the aforementioned technologies.

6.2 Data Path Chip Set Technology

Initial plans were to have the Data Path Chip Set fabricated using CMOS gate array technology, due to design time and cost factors. However, after more thorough research, CMOS field programmable gate array (FPGA) technology was chosen. An FPGA is basically a programmable gate array. High density PLDs (programmable ANDs and ORs) and PLAs (programmable ANDs, fixed ORs) also fall into the FPGA category. Conventionally, when a PLD or PLA has capacities of over 2000 gates, it can be classified as an FPGA.

CMOS FPGA technology is used for the design for several reasons. The major reason was that the design cost and design time would be greatly reduced. Designing a gate array is generally much more time consuming than designing an FPGA. A major portion of time designing a gate array is spent routing signals and simulating the device. There is also a several month lead time where the gate array is sent to a manufacturing company to be fabricated. In the event of errors, this lead time is replicated until a fully working system is developed. On FPGA systems, virtually all routing can by done automatically and simulation tends to be simpler and less critical. Simulation is less critical since incorrect designs can easily be corrected and reprogrammed. The tools used to design a gate array are much more expensive than FPGA tools. Design tools for gate arrays are on the order of \$100,000 as opposed to \$10,000 for FPGA tools, which normally include a personal computer, design / layout / simulation program, and chip programmer. Engineering costs more than double for the gate array design. For our purposes, the FPGAs will be produced in low quantities and thus are the most cost effective. Altera's MAX part FPGAs were selected as the FPGA choice, since they were the closest to fulfilling the design requirements of all the FPGA systems reviewed. A cost-effectiveness chart is shown in Figure 6.1.

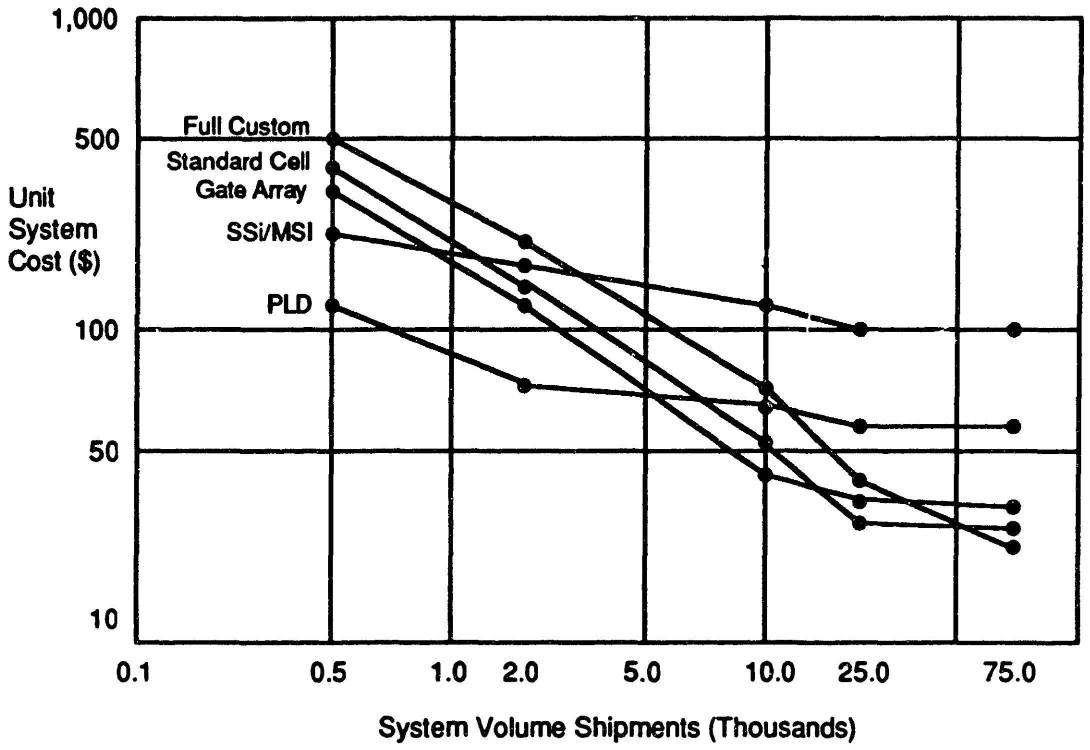


Figure 6.1 - PLD Cost Effectiveness [ALT90]

Note that the FPGAs are represented as the PLD line since Altera's FPGAs are basically large PLDs.

The problems with an FPGA system are that it tends to be slower and less compact than gate array systems. The Data Path Chip Set design was predicted to be on the order of 3500 gates with a 12.5 MHz system clock. Two FPGAs that can handle that clock rate should be able to implement the design.

6.3 Altera EPM5128 Device Characteristics

Many FPGA systems were evaluated and after much thought, the system made by Altera was chosen. The Altera system was chosen for several reasons. The Altera EPM5128 device has a high density, pin count, and speed. Each device can have the equivalent of approximately 2000 to 2500 gates, has 68 pin packages (eight of them used for power and ground), and claims a 40 MHz clock speed. Also, the Altera system had already been used at the Draper Laboratory with success.

The Altera device is based on the Logic Array Block (LAB). The EPM5128 consists of eight LABs. Each LAB consists of a number of macrocells, expander terms, and an input/output control block. The EPM5128 contains sixteen macrocells per LAB for a total of 128 macrocells. The EPM5128 has eight dedicated input pins, and 52 I/O pins. Its block diagram is shown in Figure 6.2.

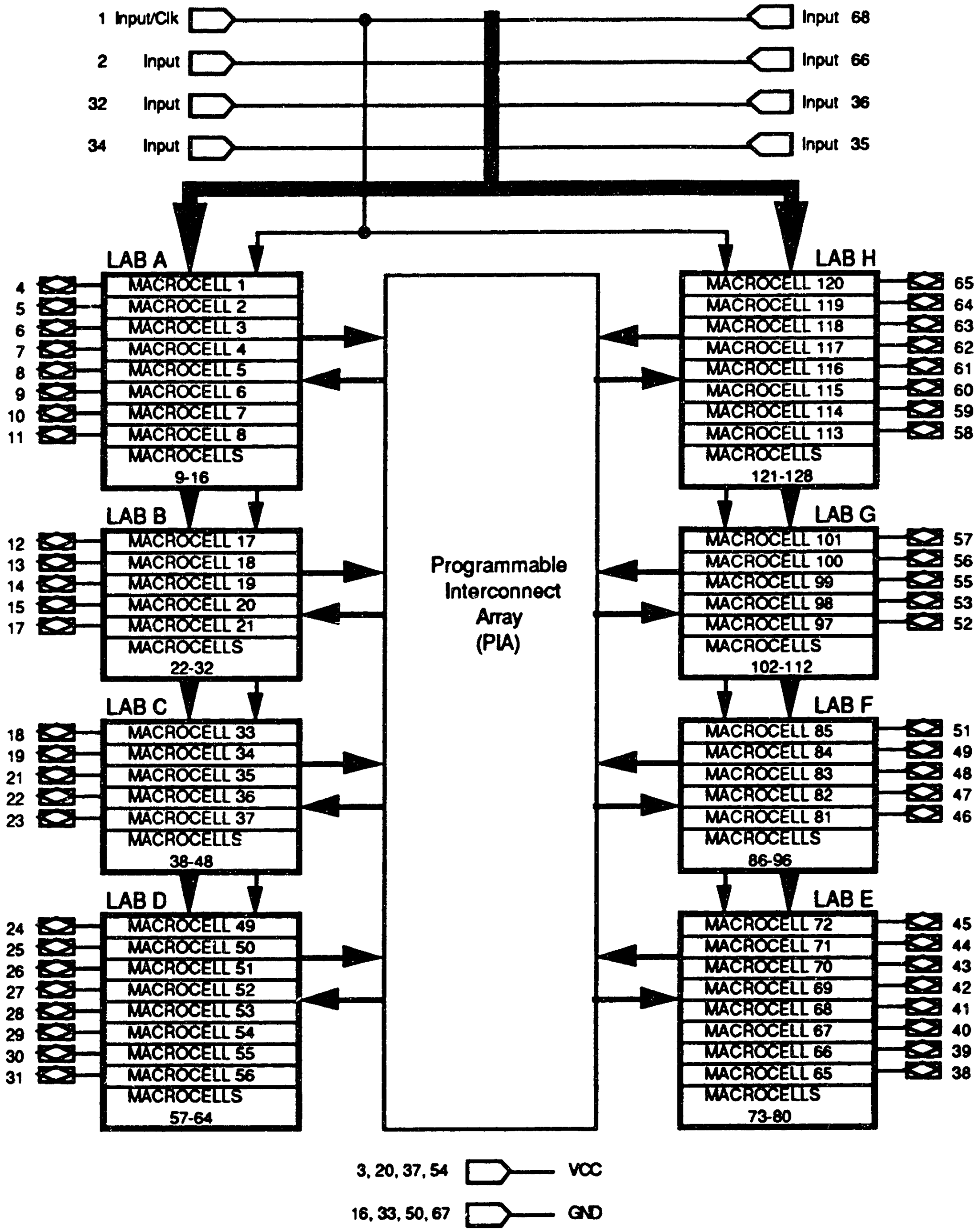


Figure 6.2 - EPM5128 Block Diagram [ALT90]

The macrocell is the basic logic element of the device. It consists of three product terms wired into a XOR gate and then into a d-flip flop register, as shown in Figure 6.3.

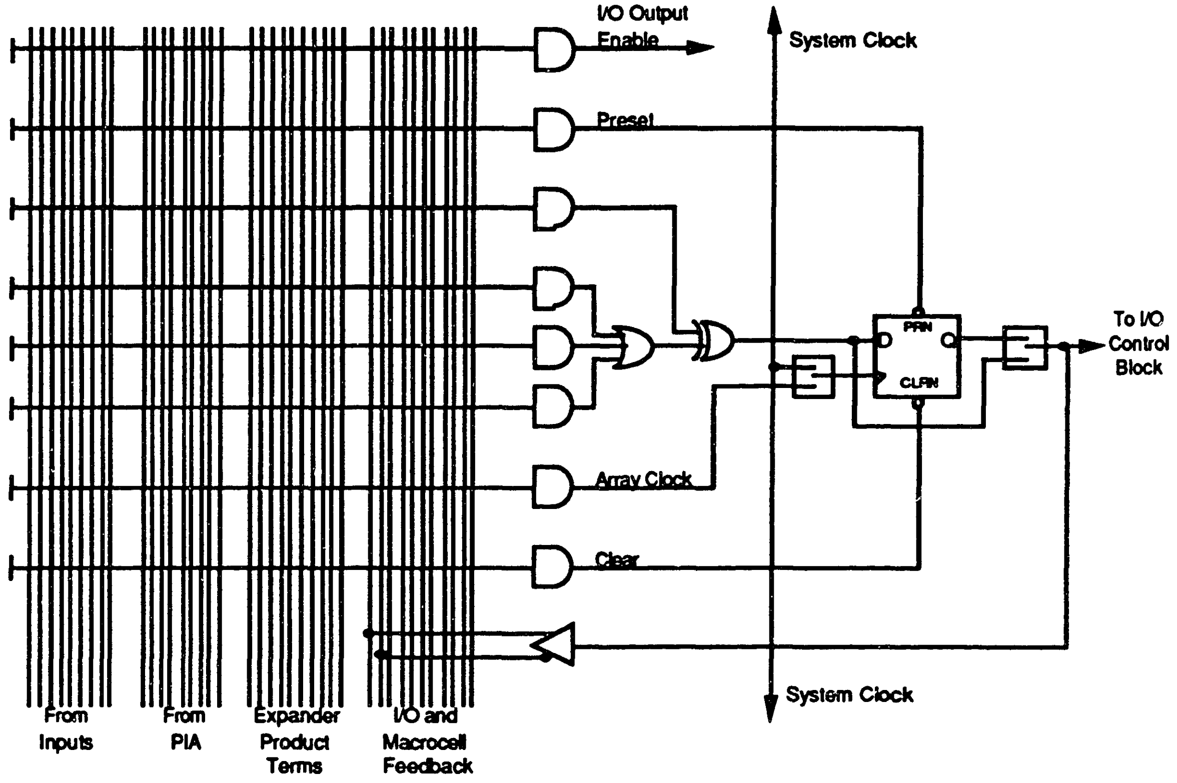


Figure 6.3 - Altera Macrocell Block Diagram [ALT90]

There are also expander terms which allow for other product terms to be included. The EPM5128 contains 32 expander terms per LAB for a total of 256 expander terms. The macrocell has a propagation delay of 25 ns and the expander term has a propagation delay of 12 ns. Therefore, when up to three product terms are needed, we can expect a propagation delay of 25 ns. When more than three product terms are needed, we can expect a propagation delay of 37 ns. Each input/output control block allows inputs to be wired into macrocells without losing the logic capability of that macrocell. However, macrocells that use the I/O pins as inputs instead of the dedicated input or feedback have a propagation delay of 40 ns. Since the majority of this design is clocked at 12.5 MHz (except for a few portions of the Fault Tolerant Clock, which are clocked at 25 MHz), we have a clock period of 80 ns. This is ample time for most of the functions necessary. The EPM5128 allows both synchronous and asynchronous clocking. Synchronous clocking is done by using the dedicated system clock (pin 1). Asynchronous clocking has a propagation delay of 14 ns, while synchronous clocking has a propagation delay of 2 ns.

The Voter FPGA uses asynchronous clocking at 12.5 MHz. The FTC FPGA uses asynchronous clocking for two clocks (12.5 MHz and an 25 MHz). The eight LABs in the EPM5128 are each interconnected by the Programmable Interconnect Array (PIA). Using the PIA, each macrocell in any LAB can use the output of any macrocell in any LAB, with a delay of 14 ns. [ALT90]

Altera recommends that power supply decoupling capacitors of at least 0.2 microfarads should be connected between each V_{CC} pin and GND, directly at the device. Also, printed circuit boards with imbedded V_{CC} and GND planes should be used to minimize system noise. [ALT90]

Since the EPM5128 is a CMOS device, power is a function of frequency and internal node switching. In order to obtain the most accurate power information, current consumption should be measured after the design is completed and placed in the system. Altera provides a typical power consumption chart as shown in Figure 6.4.

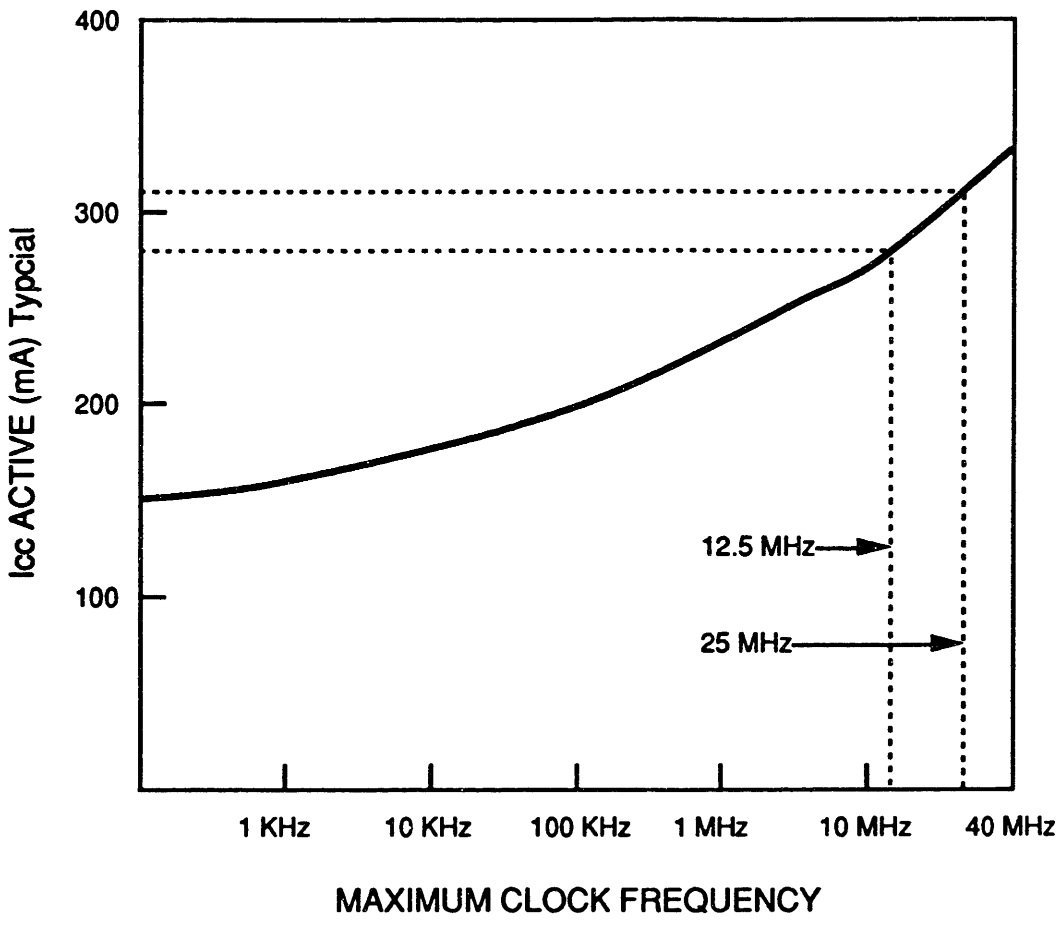
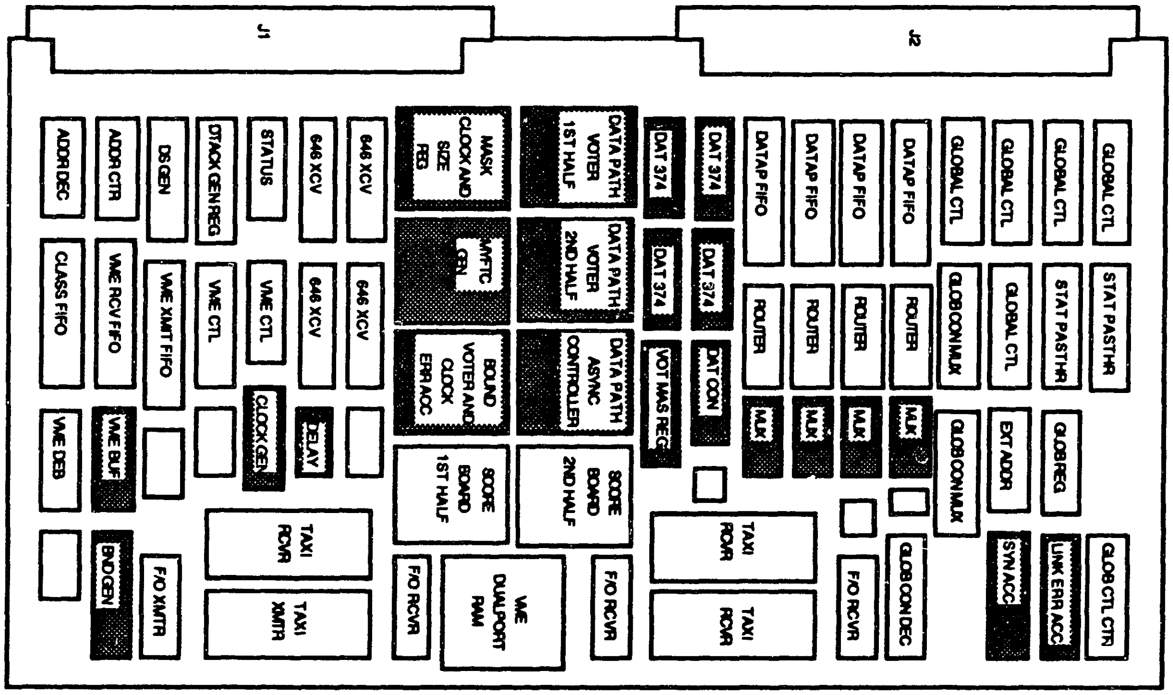


Figure 6.4 - Typical EPM5128 Power Consumption [ALT90]

The FTC chip has a maximum frequency of 25 MHz, and thus has a typical current consumption of 310 mA for a power consumption of 1.55 watts. The Voter chip has a maximum frequency of 12.5 MHz, and thus has a typical current consumption of 280 mA for a power consumption of 1.4 watts. This results in a significant power reduction over previous designs.

The EPM5128 also provides a significant component size reduction over previous designs. The EPM5128 can integrate the equivalent of 20 PAL devices or a typical mix of 75 MSI/SSI TTL chips into a single device [ALT90]. To get an idea of how much space is saved, we will take a look at the NEFTP. In Figure 6.5 we can see what parts of the NEFTP's NE could be replaced by the FTC and Voter chips.



■ Components that would be replaced by the Data Path Chip Set

Figure 6.5 - NEFTP NE Chip Equivalent

We must also take into account the fact that the C3 FTPP is much more complex than the NEFTP. The C3 FTPP has forty PEs as opposed to four in the NEFTP, and five NEs as opposed to four in the NEFTP. There are also many hardware improvements and enhancements. If the C3 FTPP equivalent for the Voter chip and FTC chips were shown in a chart such as Figure 6.5, the gray shaded area would be perhaps three times as large. Thus, the EPM5128 provides a significant space reduction.

The MAX EPM5128 technology is based on EPROM memory technology, which has proven to be highly reliable. As suggested in the previous chapter, the EPM5128s yield reduction in failures over a PAL and TTL implementation of at least three times.

Chapter 7
The Fault Tolerant Clock Chip

The Fault Tolerant Clock (FTC) Chip can be divided into six sections, the Asynchronous Control, Debug Wrap Register, Fault Tolerant Clock, Latch Decode, Link Error Accumulator, and Synchronous Control. The block diagram of the FTC Chip is shown in Figure 7.1. The Asynchronous Control section controls the loading of the receiving FIFOs, which occurs asynchronously with respect to the local NE's clock. The Debug Wrap Register wraps a byte of data from the voted data bus onto a register for future access. The Fault Tolerant Clock (FTC) synchronizes the local NE with respect to the other NEs by synchronizing itself with respect to the other NEs' message frames and generates its own message frame. The Latch Decode section is a decoder that generates signals for use throughout the FTC Chip. The Link Error Accumulator keeps track of errors that occur when a NE's clock is offsync with the other NEs' clocks. The Synchronous Control shifts data out of the FIFOs and shifts data into the CH0 FIFO, which occur synchronously to the local NE's clock.

The sections of the FTC Chip are clocked at 12.5 MHz and 25 MHz. For brevity, we will call the 12.5 MHz signal 12MHZ on the schematic diagrams. The actual pinouts for the FTC Chip are found in Appendix E.

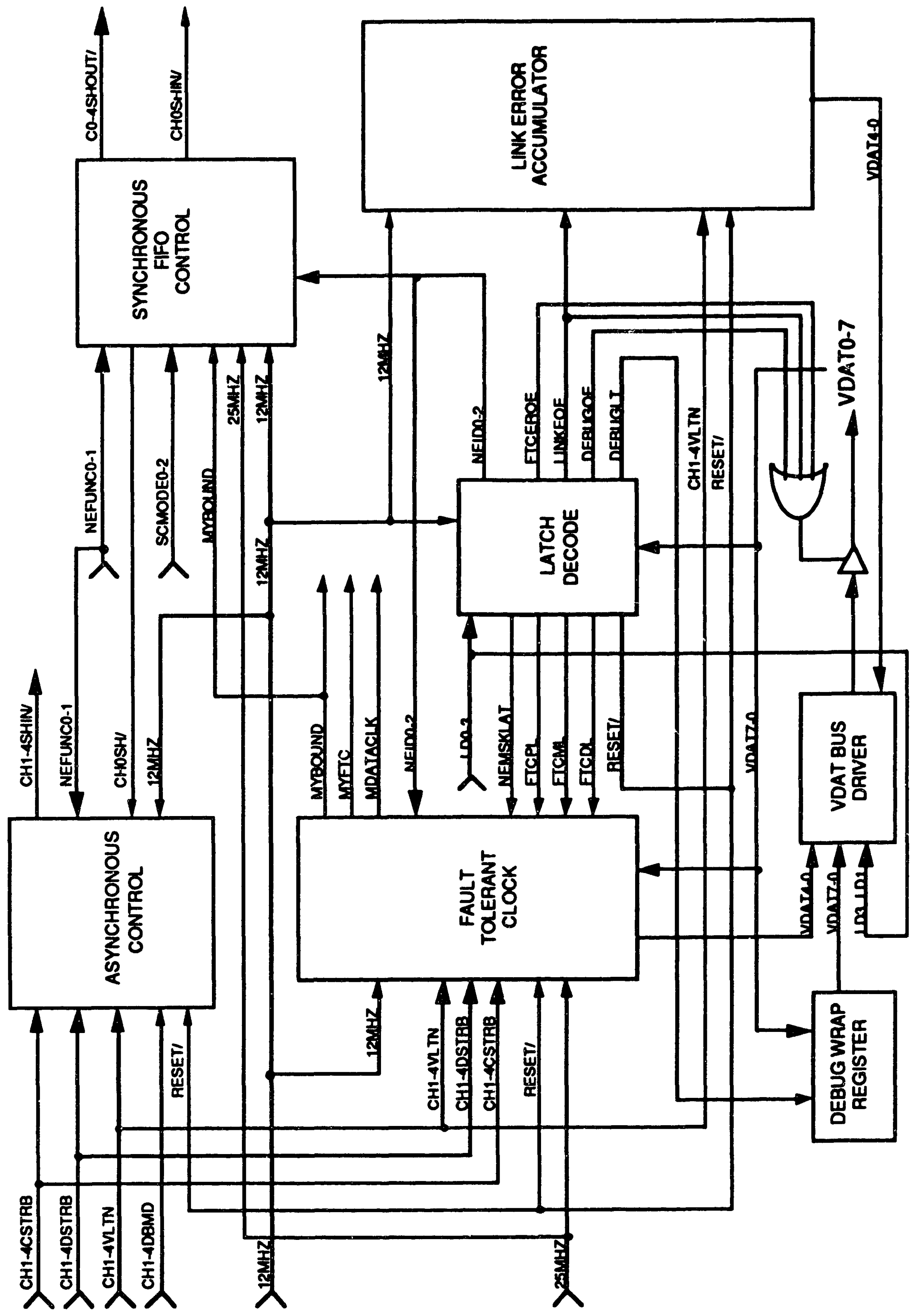


Figure 7.1 - Block Diagram of FTC Chip

7.1 Asynchronous Control

The Asynchronous Controller generates signals that are asynchronous with respect to the NE's local clock. The AMD TAXI chip signals that incoming data is valid by strobing its respective DSTRB signal high. If an error in transmission occurs, the respective CSTRB and VLTN signals are strobed high. The Asynchronous Controller uses these signals to control the loading of the CH1, CH2, CH3, and CH4 FIFOs by generating the CH1SHIN/, CH2SHIN/, CH3SHIN/, and CH4SHIN/ signals, respectively. These IDT7202 FIFOs are initiated on a high to low transition. Data is clocked in on a low to high transition with a setup time of 15 ns and a hold time of 0 ns. A byte of data is shifted in whenever any valid type of communication exchange is taking place. NEFUNC1 and NEFUNC0 determine what type of exchange is taking place as shown in Table 7.1. NEFUNC1 and NEFUNC0 are initiated on the rising edge of MYFTC, from the Fault Tolerant Clock section. Data is shifted in when the NE is in Data XMIT, Data Reflect, or SERP mode. Data should not be shifted in when the NE is in Idle mode since non-valid messages are sent for synchronization purposes.

NEFUNC1	NEFUNC0	Type of Exchange
0	0	Idle
0	1	Data XMIT
1	0	Data Reflect
1	1	SERP

Table 7.1 - Asynchronous Control Exchange Modes

A byte of data is shifted in during a valid communication mode when one of three cases happen. In the first case, data is shifted into a FIFO when the appropriate data strobe from a receiving TAXI chip goes high (nDSTRB), indicating that data is ready to be read from an incoming FIFO. In the second case, data is shifted in when an error occurs (nCSTRB and nVLTN go high). This measure is taken so that all the message bytes are shifted even during transmission errors. In the third case, there is a debug mode for each channel, which is enabled by the CH1DBMD, CH2DBMD, CH3DBMD, and CH4DBMD signals from the NE Global Controller. When these modes are enabled, data should be shifted in straight from the output of the Voter chip. This allows data to be wrapped back to the NE. When in debug mode, the SHIN/ signals become the CH0SHIN/ signal from the Synchronous Control section in a asynchronous form (CH0SH/). The logic equations for the shifting in of data are as follows, where n stands for CH1, CH2, CH3, or CH4, representing the Channel 1, 2, 3, or 4 FIFOs, respectively.

nSHIN/ = nDSTRB • (NEFUNC0 + NEFUNC1) • nDBMD/
+ nCSTRB • nVLTN • (NEFUNC0 + NEFUNC1) • nDBMD/
+ CH0SH/ • nDBMD

7.2 Debug Wrap Register

The VDAT bus, which stands for the voted data bus, transfers various types of information throughout the NE. The Debug Wrap Register is an eight-bit-wide register that holds data from the VDAT bus so that it may be read by the NE at a later time. Note that the Debug Wrap Register has no relevance to the debug mode that is used in the Asynchronous Control described in the previous section. A byte of data is latched on the Debug Wrap Register by the DEBUGLT signal from the Latch Decode section. A byte of data is put onto the VDAT data bus by the DEBUGOE signal, also from the Latch Decode section. The Debug Wrap Register consists of combinational logic followed by synchronous latches as shown for one bit in Figure 7.2, where <n> represents the bits 0 through 7.

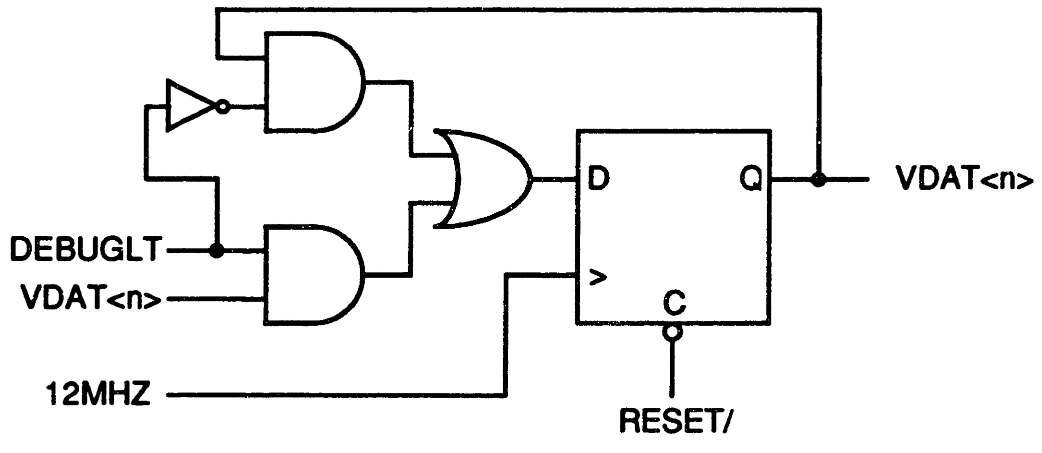


Figure 7.2 - Debug Wrap Register Circuitry

7.3 Fault Tolerant Clock

The Fault Tolerant Clock (FTC) is the mechanism that synchronizes the NEs and records synchronization errors. It is the bulk of the FTC Chip. The synchronization takes place through message exchanges as follows. The Global Controller tells the FTC the length, in bytes, of the next message frame, called a packet. The FTC generates the signals MYBOUND, and MYFTC. MYBOUND is the frame boundary in which communication takes place. Synchronization is be achieved by synchronizing MYBOUND with respect to the BOUND signals from the other NEs. MYBOUND is also MYFTC delayed by one clock cycle. A timing diagram of the FTC synchronization mechanism is shown in the Simulation chapter in Section 9.2.3 and in Appendix C. MYFTC is used by the NE for transmission of data via the fiberoptic transmitter.

The FTC can be further divided into the Bane State Machine, the Bound Generator, the Mask Latch, the Mid Data Transfer, My State Machine, the Packet Latch and Counter, the Presence Accumulator, the Propagation Delay Latch and Counter, and the Synchronization Feedback sections. The Bane State Machine calculates whether or not the NE's message frame is behind or ahead of the other NE's message frames. The Bound Generator generates BOUND signals from the other NE's data clocks, which indicate their respective message frames. The Mask Latch holds the NE masks, which determine which NEs are in operation. The Mid Data Transfer, as the name suggests, indicates the middle of the data transfer. My State Machine generates MYBOUND and MYFTC, loads the packet count, and loads the propagation delay count. The Packet Latch and Counter holds the packet length and counts down. The Presence Accumulator accumulates presence errors of the clocks. The Propagation Delay Latch and Counter holds inter-NE propagation delay and indicates the start of Bane State Machine. The Synchronization Feedback returns the Bane State Machine signals to the My State Machine in a different format. The block diagram for the FTC is shown in Figure 7.3.

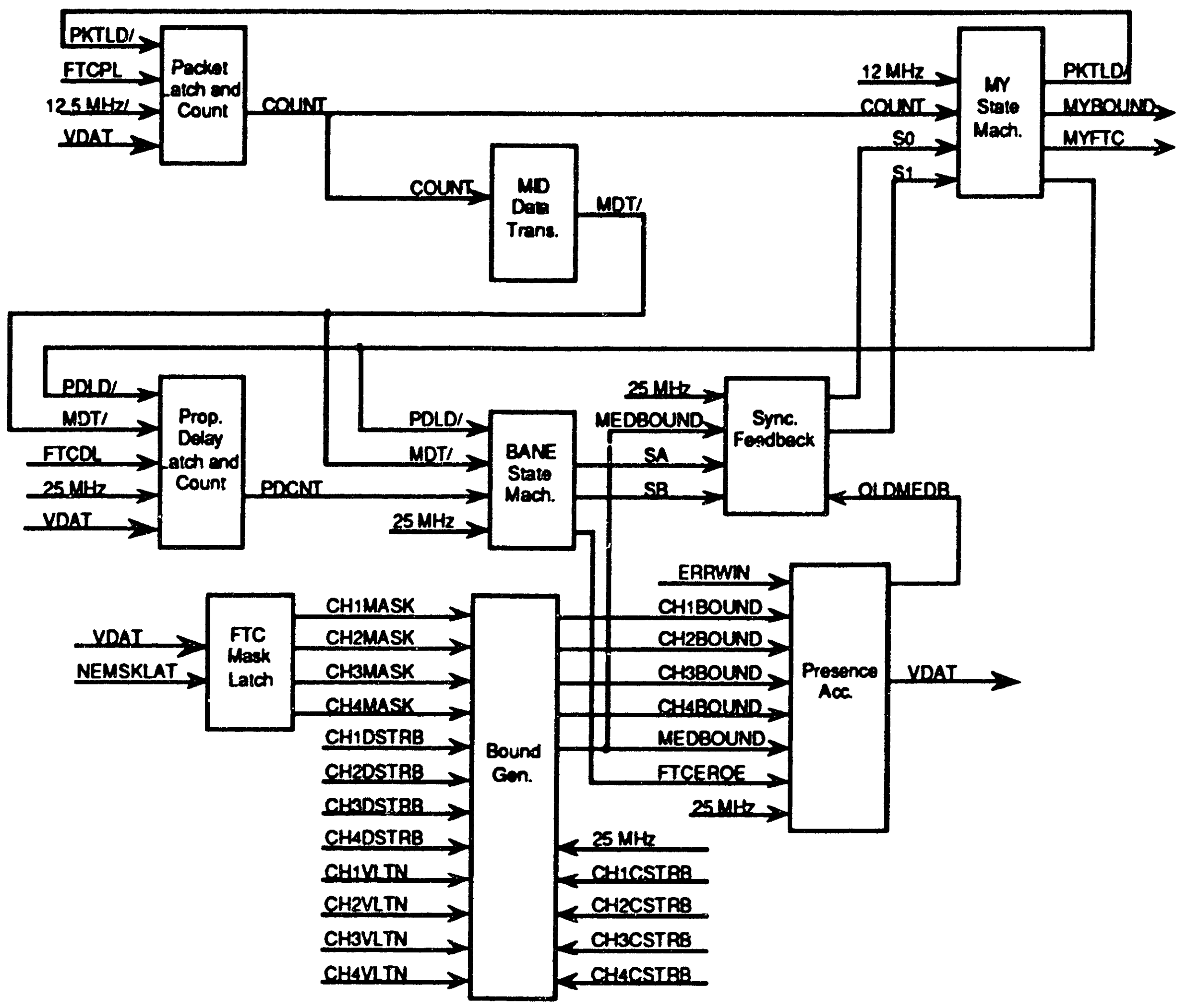
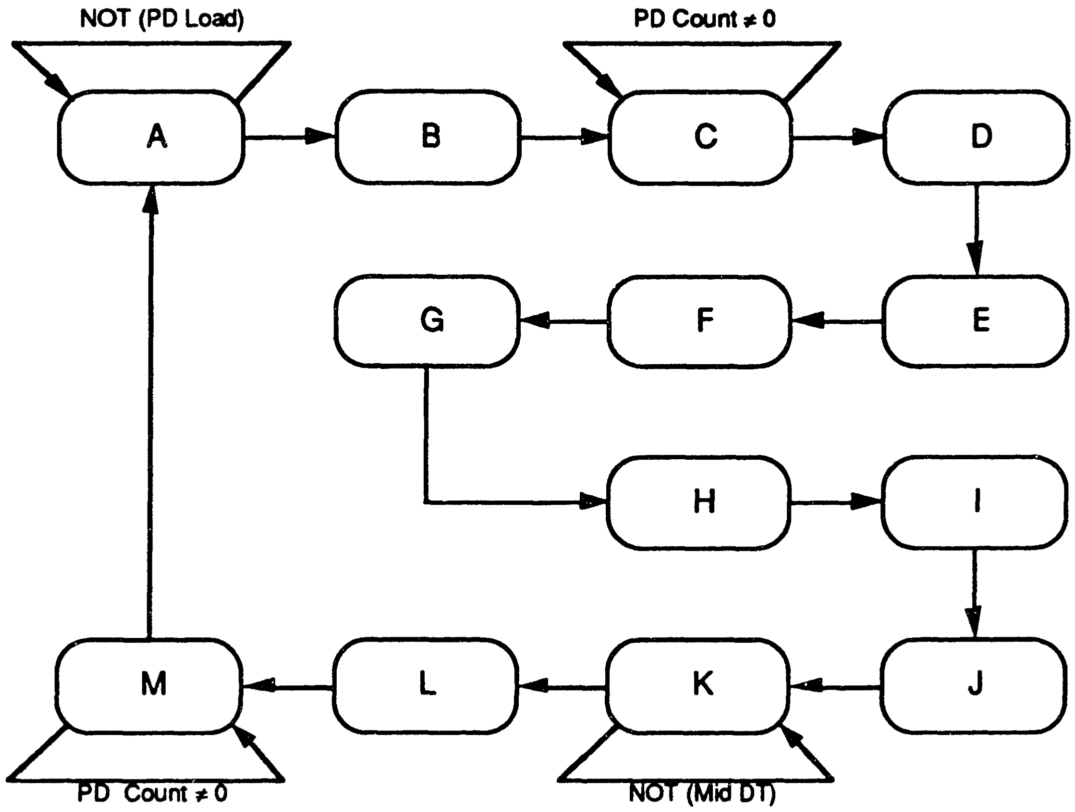


Figure 7.3 - FTC Block Diagram

7.3.1 Bane State Machine

The Bane (Behind/Ahead/Normal/Error) State Machine calculates whether or not the NE's message frame is behind or ahead of the other NE's message frames. This is part of the NE synchronization mechanism. It is triggered on the middle of PDLD/ (from the Bane State Machine) and MDT/ (from the Mid Data Transfer). It generates ERRWIN (indicating the error window), SAW (the self-ahead window), and SBW (the self-behind window). ERRWIN indicates whether or not a channel is off-sync with respect to the other channels. ERRWIN is low for three and a half clock cycles (280 ns). By comparing ERRWIN to the BOUND signals from the Bound Generator, we can determine which NEs are off-sync. SAW indicates that a channel's FTC is at least one and a half clock cycles (120 ns) ahead of the other channel's FTCs. SBW indicates that a channel's FTC is at least one and a half clock cycles (120 ns) behind the other channel's FTCs. If a channel's FTC is neither ahead or behind, then it should be synchronized to the other channels' FTCs to within one and a half clock cycles (120 ns). The synchronization adjustment is done by the My State Machine, which uses the Bane State Machine signals from the Synchronization Feedback. The state diagram for the Bane State Machine is shown in Figure 7.4. The Bane State Machine is clocked at 25 MHz. A timing diagram of the Bane State Machine is shown in Section 9.2.3.



State	Binary	ERRWIN	SBW	SAW
A	110000	1	1	0
B	110010	1	1	0
C	110001	1	1	0
D	010001	0	1	0
E	010000	0	1	0
F	000001	0	0	0
G	000000	0	0	0
H	000011	0	0	0
I	001000	0	0	1
J	001001	0	0	1
K	101000	1	0	1
L	101010	1	0	1
M	101100	1	0	1

Figure 7.4 - State Diagram for Bane State Machine

The state equations for the Bane State Machine are shown in Table 7.2. The first term in the Q2 function is used to remove a trap state. The state for the Bane State Machine were chosen so that the state equations were limited to three product terms. For the actual layout of the Bane State Machine in the Altera chip, this eliminates the need for expanders, and allows the Bane State Machine to run at 25 MHz.

$$Q5 := Q5/\cdot Q3\cdot Q0 + Q5\cdot Q0/ + Q5\cdot Q0\cdot(\text{Propagation Delay} \neq 0)$$
$$Q4 := Q5\cdot Q4 + Q4\cdot Q0 + Q2\cdot(\text{Propagation Delay} = 0)$$
$$Q3 := Q1\cdot Q0 + Q4/\cdot Q3\cdot Q2/ + Q2\cdot(\text{Propagation Delay} \neq 0)$$
$$Q2 := Q5\cdot Q4/\cdot Q3/\cdot Q2/\cdot Q1/\cdot Q0/ + Q3\cdot Q1 + Q2\cdot(\text{Propagation Delay} \neq 0)$$
$$Q1 := Q5/\cdot Q4/\cdot Q3/\cdot Q0 + Q5\cdot Q4\cdot Q1/\cdot Q0/\cdot PDLD + Q5\cdot Q3\cdot Q2/\cdot Q1/\cdot MDT$$
$$Q0 := Q4\cdot Q1 + Q5\cdot Q0 + Q5/\cdot Q0/$$
$$ERRWIN = Q5$$
$$SBW = Q4$$
$$SAW = Q3$$

Table 7.2 - Bane State Machine State Equations

7.3.2 Bound Generator

The Bound Generator generates BOUND signals from the other NE's data clocks to generate CH1BOUND, CH2BOUND, CH3BOUND, and CH4BOUND. The BOUND signals are generated by taking the frame boundaries of the respective DSTRB, CSTRB, and VLTN signals. The BOUND signals are generated by combinational logic which feeds set-reset (SR) latches, then sampled at 25 MHz by a D flip-flop as shown in Figure 7.5. Since the DSTRB, CSTRB, and VLTN signals are asynchronous, it is possible that the BOUND signals may enter a metastable state. However, these BOUND signals are fed into the Presence Accumulator (see Section 7.3.7), which also contains D flip-flops. Thus, the possibility of metastability is reduced.

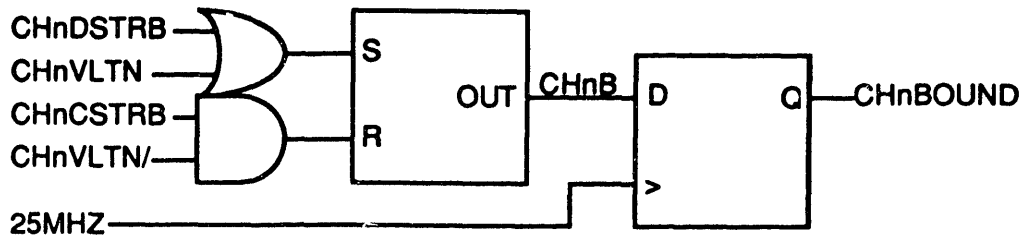


Figure 7.5 - Bound Generation Schematics

The Bound Generator then takes these four incoming channel BOUND signals and outputs the second BOUND signal received as the median bound (MEDBOUND). This is accomplished by a modified version of the edge-triggered clock voter used in the FPMP. [TBS81] The MEDBOUND generation logic is shown in Figure 7.6.

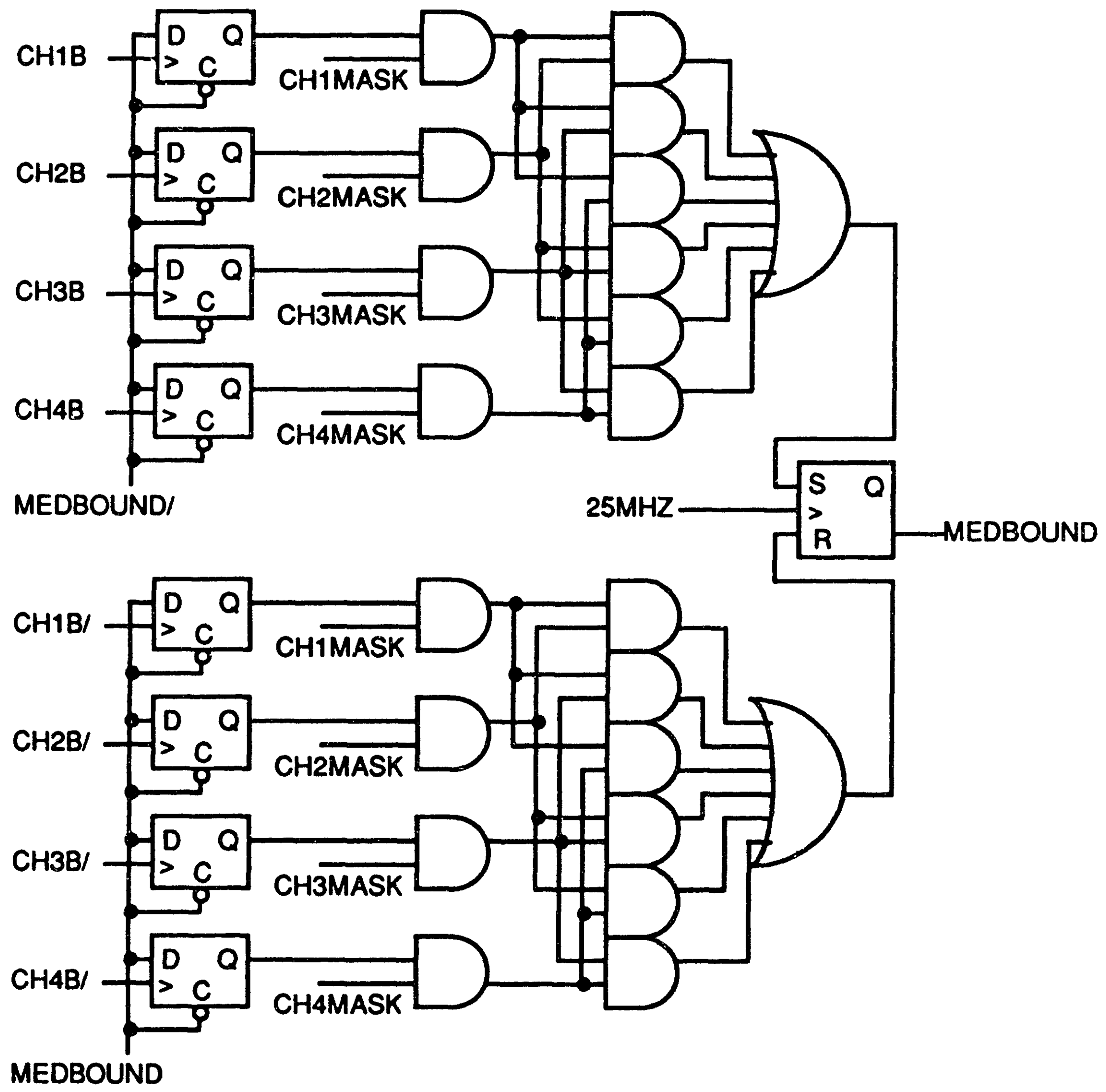


Figure 7.6 - MEDBOUND Generation

7.3.3 Mask Latch

The FTC Mask Register provides masks for the Fault Tolerant Clock, indicating which NEs are operational. These masks are used to mask out BOUND signals from faulty NEs. The masks are active high, representing non-faulty NEs, and are latched from the low five bits of the VDAT bus by NEMSKLAT from the FTC Latch Decode section. The VDAT bits are latched as shown in Table 7.3. Note that the masks are latched in absolute format.

VDATBIT	NEMASKLATCH
5-7	Unused
4	NEEMASK
3	NEDMASK
2	NECMASK
1	NEBMASK
0	NEAMASK

Table 7.3 - VDAT NE Mask Assignments

The NE Masks are converted from absolute form (NEA-EMASK) to relative form (CH0-4MASK) by the mechanism shown in Figure 7.7. It is basically a five-bit-wide, five-to-one multiplexer selected by the NEID signals from the FTC Latch Decode section.

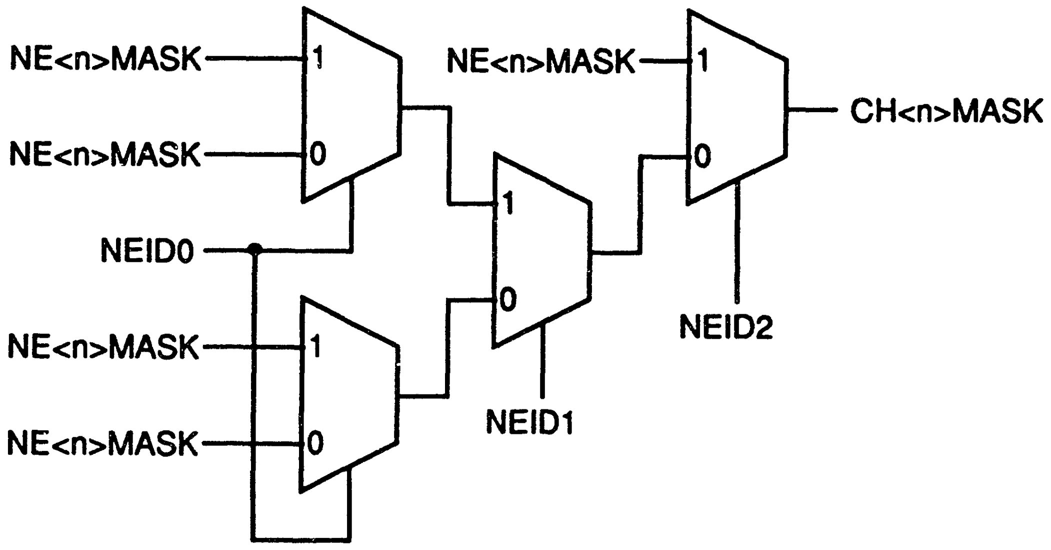


Figure 7.7 - Absolute to Relative Conversion

The masks are then latched by NEMSKLAT, as shown in Figure 7.8.

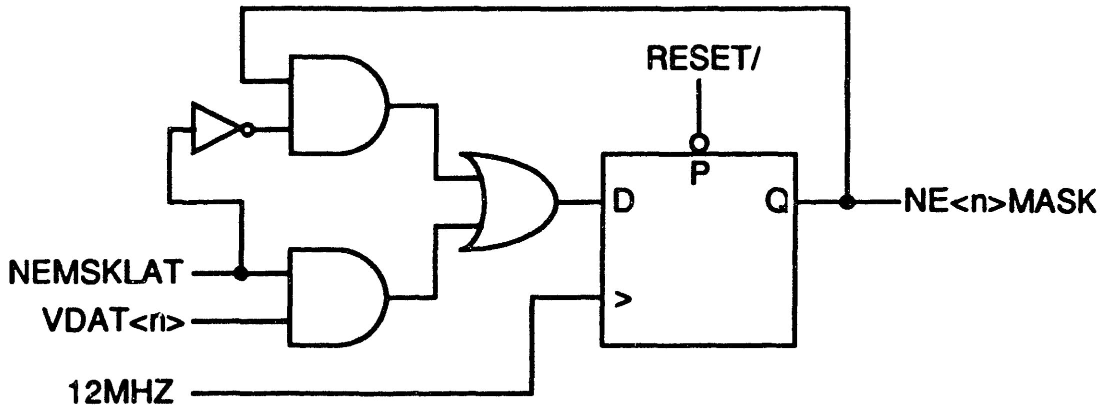


Figure 7.8 - Mask Latch Circuitry

7.3.4 Mid Data Transfer

The Mid Data Transfer section indicates the middle of the data transfer frame. The MDT/ signal is strobed low when the packet count reaches half its value. The MDT/ signal is sent to the Bane State Machine to offset the error windows if necessary and to the Propagation Delay Latch and Counter to initiate a new propagation delay count. MDT/ is generated by XOR gates as shown in Figure 7.9.

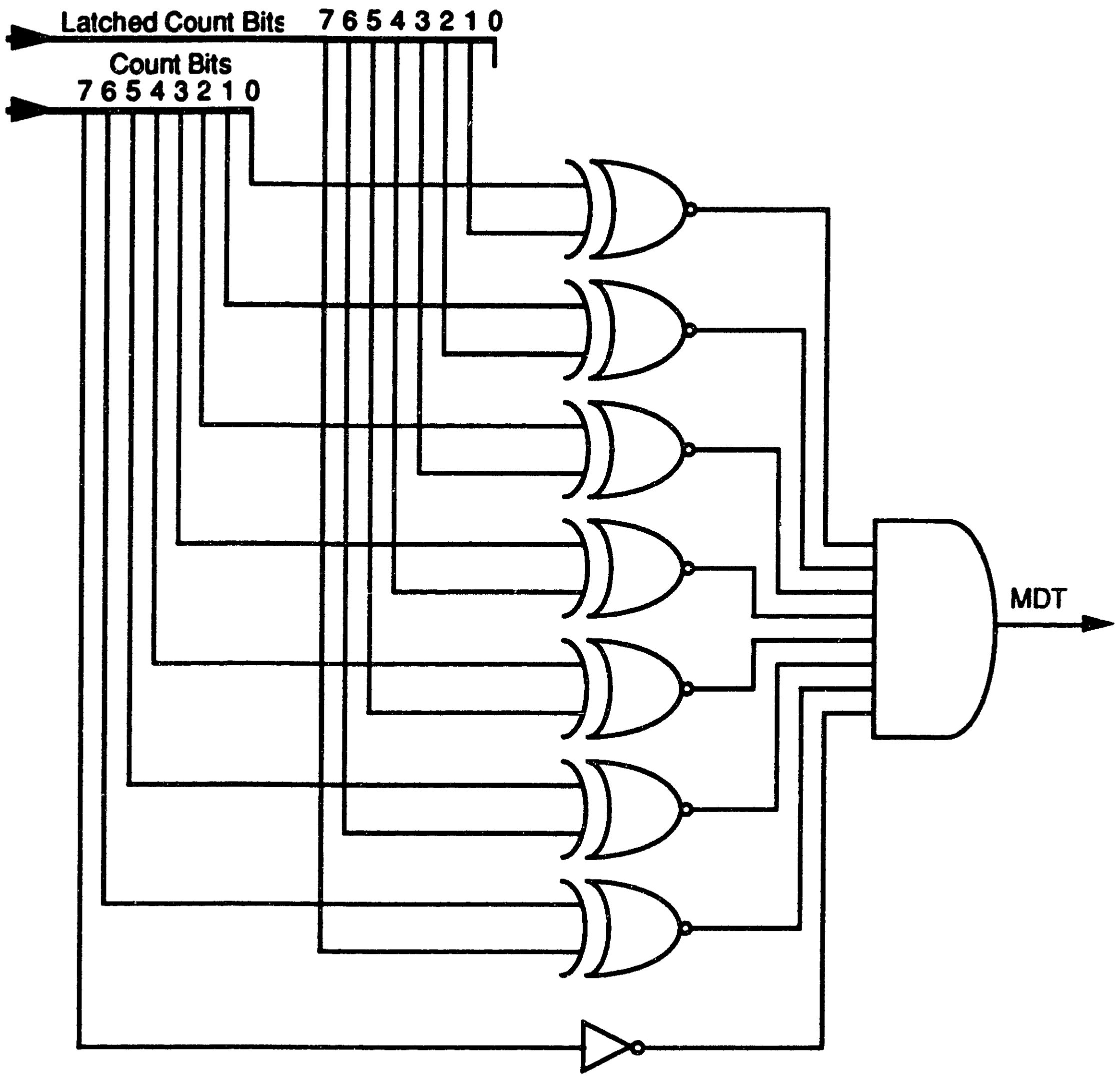
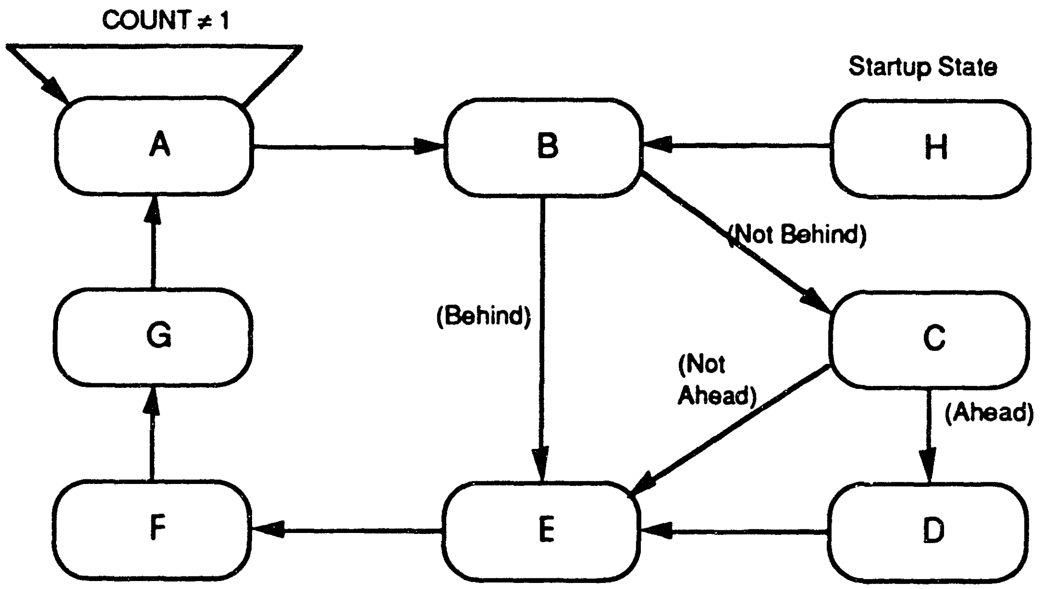


Figure 7.9 - Mid Data Transfer Logic

7.3.5 My State Machine

My State Machine generates the MYBOUND and MYFTC signals using the state machine shown in Figure 7.10. As said previously, MYBOUND is the frame boundary in which communication takes place. MYFTC is used to generate MYDATCLK, which transmits data to the other NEs. There are normally four clock cycles (320 ns) in between MYBOUND signals. However, the My State Machine adjusts the MYBOUND and MYFTC signal when the NE is ahead or behind the other NEs. This is the mechanism that does the actual adjusting of a local NE's clock with respect to the other clocks. When the local NE is ahead (as determined from signals from the FTC Synchronization Feedback section, My State Machine puts five clock cycles in between MYBOUND signals. When the local NE is behind, My State Machine puts three clock cycles in between MYBOUND signals. MYBOUND is MYFTC delayed by one clock cycle. Since a NE's clock can be adjusted by one clock cycle per message frame, it can be estimated that synchronization should be guaranteed after n/2 message frames, where n is the packet count. There is a small chance that clocks may start up in a metastable state where clocks are chasing each other, but this chance is small (on the order of 10^{-6} and only during startup with four NEs) and should reach synchronization as time passes. My State Machine also generates PKTLD/, which loads the packet count, and PDLD/, which loads the propagation delay. A timing diagram of the My State Machine is shown in Section 9.2.3.



State	Decimal	Binary	MYFTC	MYBOUND	PKTLD	PDLD
A	12	1100	1	1	0	0
B	4	0100	0	1	0	0
C	2	0010	0	0	0	0
D	1	0001	0	0	0	0
E	0	0000	0	0	0	1
F	3	0011	0	0	0	0
G	8	1000	1	0	1	0
H	7	0111	0	1	0	0

Figure 7.10 - State Diagram for My State Machine

The state equations for My State Machine are shown in Table 7.4.

Q3 := Q2/•Q1•Q0 + Q3•Q2/ + Q3•Q2•(COUNT ≠ 1)
Q2 := Q3 + Q2•Q1•Q0
Q1 := Q3/•Q2•Q1/•S1/ + Q3/•Q2/•Q1/•Q0/
Q0 := Q1•Q0/•S0 + Q3/•Q2/•Q1/•Q0/
MYFTC = Q3
MYBOUND = Q2
PKTLD := Q2/•Q1•Q0
PDLD := Q3/•Q2•Q1/•S1 + Q1•Q0/•S0/ + Q1/•Q0

Table 7.4 - My State Machine State Equations

7.3.6 Packet Latch and Counter

The Packet Latch and Counter holds the packet length and counts down. It halts the FTC if the packet length reaches zero. The packet count is latched from the VDAT bus when FTCPL is enabled from the latch decode by the mechanism shown in Figure 7.11.

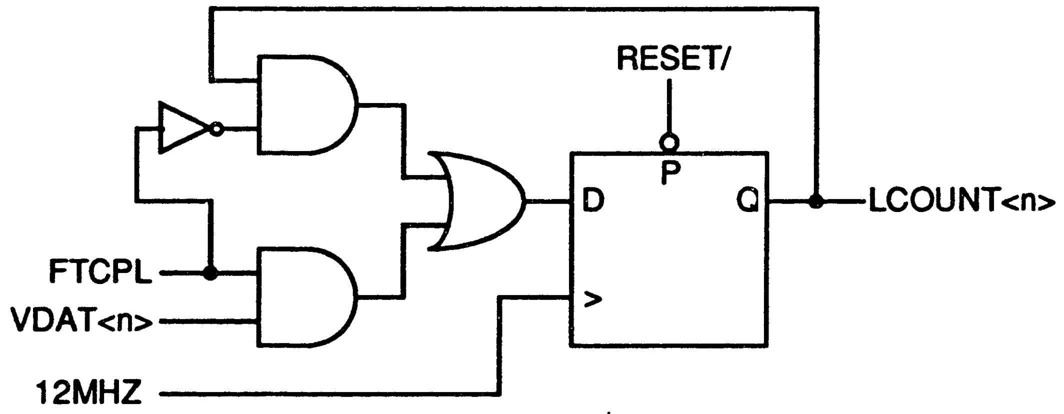


Figure 7.11 - Packet Latch Schematics

The latched count is then sent to a pair of four-bit down counters that are basically modified 74161 counters, as shown in Figure 7.12. The count is reset to the latched count when PKTLD/ is strobed.

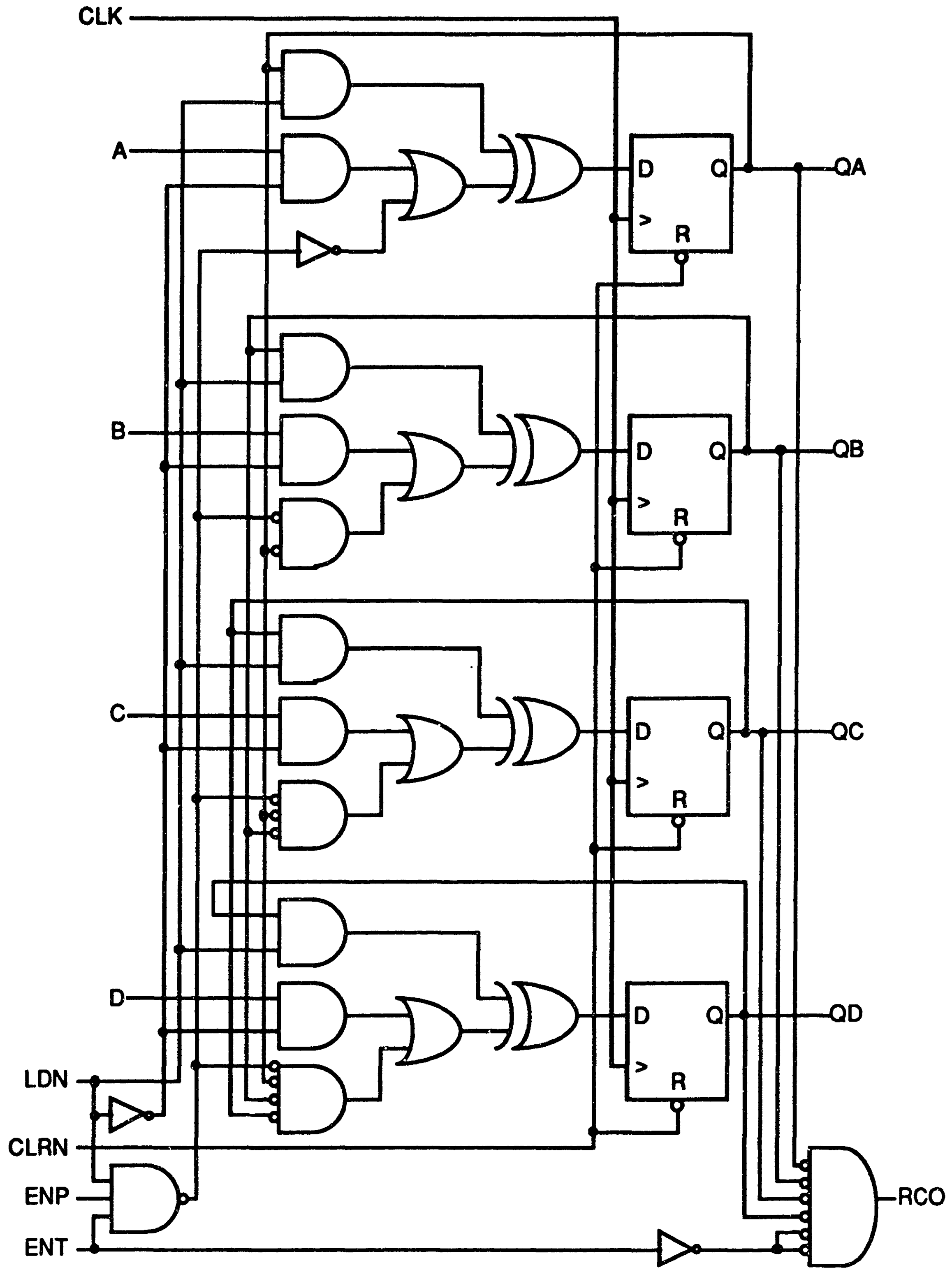


Figure 7.12 - Four-Bit Down Counter

7.3.7 Presence Accumulator

The Presence Accumulator accumulates presence, or synchronization, errors of the clocks. A presence error is asserted high when a NE's FTC is off sync with the other NEs by more than two and a half clock cycles (200 ns). The error information is written to the VDAT bus when FTCEROE is selected from the latch decode in the format shown in Table 7.5.

VDATBIT	PRESENCE ERROR
5- 7	Set Low
4	CH 4 Error
3	CH 3 Error
2	CH 2 Error
1	CH 1 Error
0	Set Low

Table 7.5 - Presence Accumulator Output Bits

Note that CH 0 errors are not recorded by the Presence Accumulator. This is because an NE cannot mask itself out in terms of synchronization. The Bound Generator generates MEDBOUND from the four incoming BOUND signals, so masking the local NE would serve no purpose for synchronization. Presence errors occur when a channel's BOUND signal rises outside of the error window. This is accomplished by the mechanism shown in Figure 7.13.

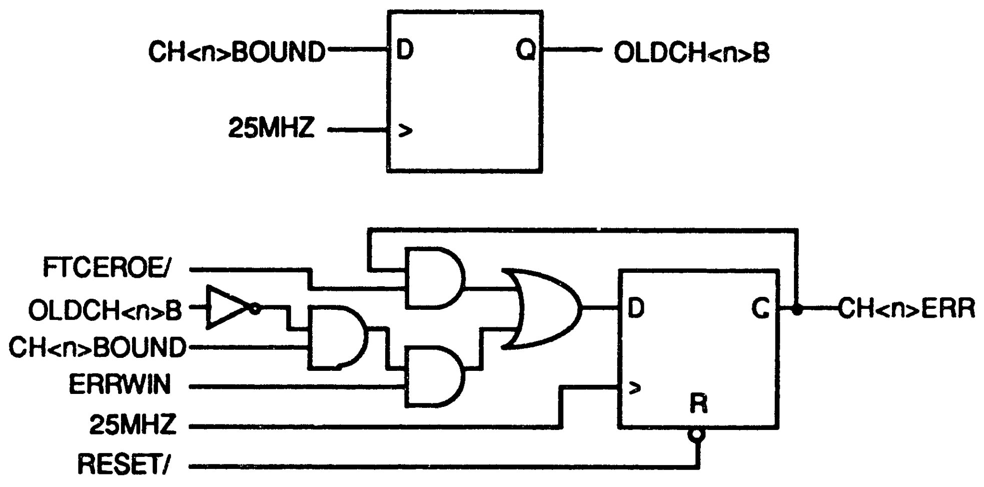


Figure 7.13 - Presence Accumulator Schematics

7.3.8 Propagation Delay Latch and Counter

There are propagation delays between NEs that result due to the delay of information transfer between NEs. The Propagation Delay Latch and Counter holds inter-NE propagation delay and indicates the start of Bane State Machine. The low six bits of the VDAT bus (VDAT5-0) are latched on as the propagation delay when FTCDL is enabled from the latch decode. The propagation delay is clocked at 25 MHz, so each propagation delay count represents a delay of 40 ns. The propagation delay is latched from the VDAT bus when FTCDL is enabled from the latch decode by the mechanism shown in Figure 7.14.

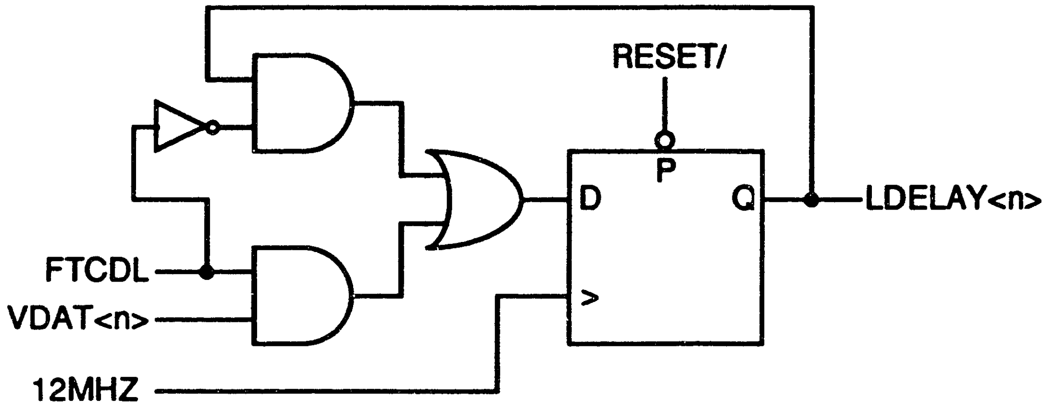


Figure 7.14 - Propagation Delay Latch Schematics

The latched delay is then sent through a pair of four-bit down counters that are identical to the modified 74161 counters shown in Figure 7.12. The count is reset to the latched count when PDLD/ or MDT/ are strobed.

7.3.9 Synchronization Feedback

Synchronization Feedback takes the self-ahead window (SAW) and self-behind window (SBW) signals and determines whether the system's clock is ahead, behind, or normal with respect to the median bound (MEDBOUND). The signals are sent to the My State Machine as shown in Table 7.6. The function of the Synchronization Feedback is to latch the correct synchronization status of the local NE for use by several states of the My State Machine. Simply using the self-ahead and self-behind windows (SAW and SBW) would yield incorrect results.

S1	S0	Condition
0	1	Ahead
0	0	Normal
1	1 or 0	Behind

Table 7.6 - Synchronization Feedback Signals

The Synchronization Feedback signals are generated by the logic shown in Figure 7.15. It basically latches SAW and SBW on the rising edge of MEDBOUND.

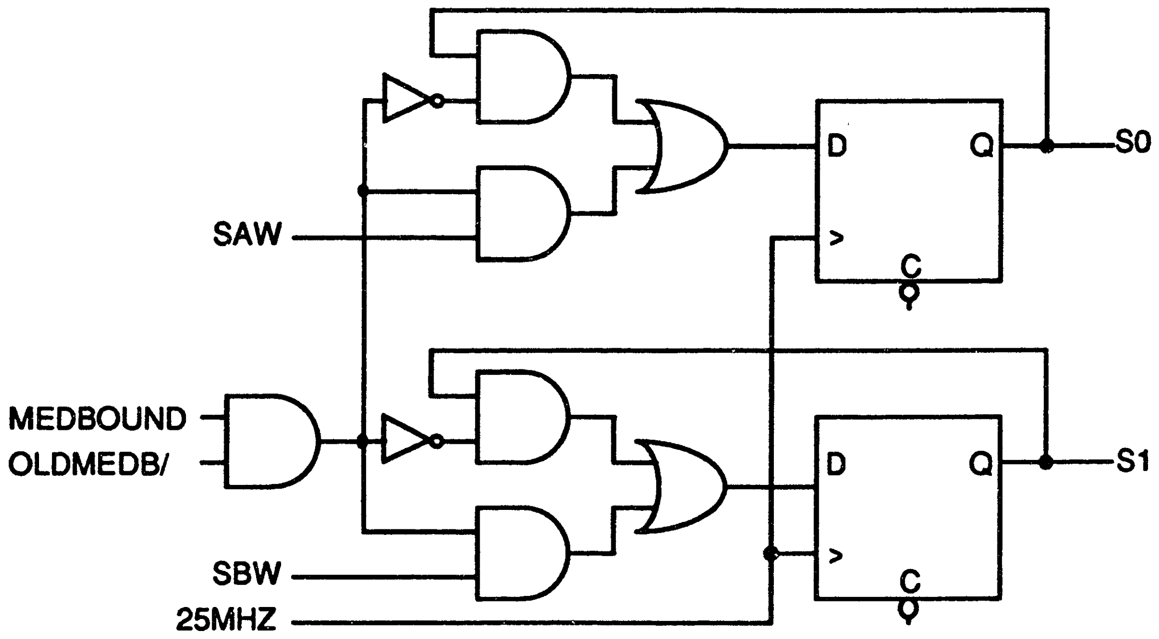


Figure 7.15 - Synchronization Feedback Schematics

7.4 Latch Decode - FTC

The VDAT bus carries data to and from many places in the gate array. Many control signals are needed to latch data from the VDAT bus into registers and to enable data from registers onto the VDAT bus. To conserve the gate array I/O, a group of four signals are decoded to provide all of the data latches and enables for the registers connected to the VDAT bus. LD0, LD1, LD2, and LD3 are the four Latch Decode signal inputs. They are decoded as shown in Table 7.7 for the FTC chip.

LD3	LD2	LD1	LD0	RESULTING OUTPUT
0	0	0	0	Idle (no output)
0	0	0	1	Reset the system (RESET/)
0	0	1	0	NEID Latch (NEIDLAT)
0	0	1	1	FTC Packet Latch (FTCPL)
0	1	0	0	FTC Delay Latch (FTCDL)
0	1	0	1	FTC Error Output Enable (FTCEROE)
0	1	1	0	Link Error Output Enable (LINKEOE)
0	1	1	1	NE Mask Reg. Latch (NEMSKLAT)
1	1	0	0	Debug Wrap Reg. Latch (DEBUGLT)
1	1	0	1	Debug Wrap Reg. Enable (DEBUGOE)

Table 7.7 - FTC Latch Decode Signals

RESET/ resets the FTC chip. NEIDLAT latches the VDAT2-0 signals to become the Network Element Identification (NEID) signals NEID2-0, respectively. The NEID signals indicate the absolute value of the NE as shown in Table 7.8.

NE	NEID2	NEID1	NEID0
A	0	0	0
B	0	0	1
C	0	1	0
D	0	1	1
E	1	0	0

Table 7.8 - NE Identification Signals

FTCPL latches the packet count from VDAT 7-0 (see Section 7.3.6). FTCDL latches the delay count from VDAT5-0 (see Section 7.3.8). FTCEROE puts the FTC errors that have occurred onto the VDAT bus (see Section 7.3.7) on the clock cycle after the Latch Decode signals are enabled and resets the presence errors. LINKEOE puts the link errors that have occurred onto the VDAT bus on the clock cycle after the Latch Decode signals are enabled (see Section 7.5). NEMSKLAT latches the VDAT4-0 signals as the FTC masks (see Section 7.3.4). DEBUGLT latches data on the VDAT bus onto the Debug Wrap Register and DEBUGOE puts this data back onto the VDAT bus on the clock cycle after the Latch Decode signals are enabled (see Section 7.2). The Latch Decode is a basic decoder built out of combinational logic and registers. The RESET/ signal is registered. All the other Latch Decode signals are purely combinational.

7.5 Link Error Accumulator

The Link Error Accumulator records any transmission errors that result from message exchanges. A rising edge of the VLTN signals from an AMD TAXI chip indicates that a transmission error was detected. This VLTN signal is followed by either a DSTRB signal or CSTRB signal. Transmission errors are recorded by latching the VLTN signals from the four incoming AMD TAXI chips. A high signal indicates an error has occurred. This information is requested from the Global Controller when needed. The information is written to the VDAT bus as shown in Table 7.9.

VDATBIT	LINKERROR
5- 7	Set Low
4	CH 4 Error
3	CH 3 Error
2	CH 2 Error
1	CH 1 Error
0	Set Low

Table 7.9 - Link Error Accumulator Output Bits

The link error signals, CH0VLTN, CH1VLTN, CH2VLTN, CH3VLTN, and CH4VLTN, are sent through a pair of latches and output as CH1LERR, CH2LERR, CH3LERR, and CH4LERR, respectively. They are output to the VDAT bus by the

LINKEOE signal. Link errors are stored on the Link Error Accumulator until the LINKEOE signal or RESET/ is strobed. This Link Error Accumulator mechanism is shown in Figure 7.16.

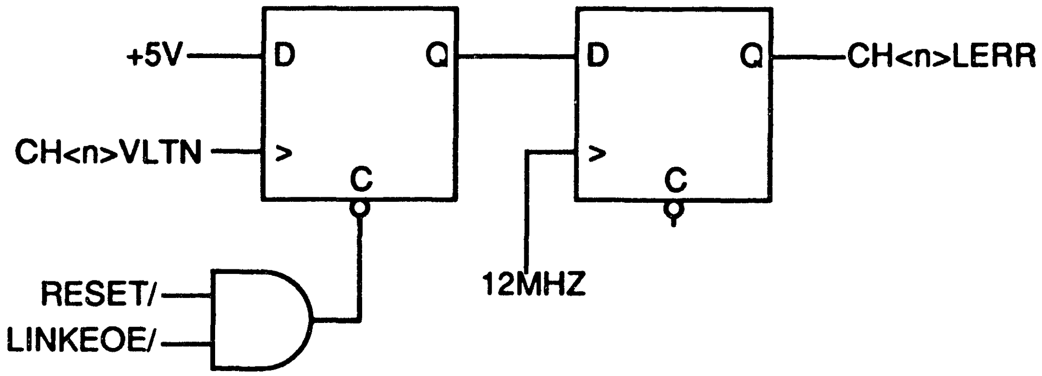


Figure 7.16 - Link Error Accumulator Schematics

7.6 Synchronous Control

The Synchronous Control section of the FTC chip controls those signals that occur synchronously with respect to the local NE's clock. This includes signals which shift data out of each of the FIFO's (CH0SHOUT/, CH1SHOUT/, CH2SHOUT/, CH3SHOUT/, and CH4SHOUT/). It also generates CH0SHIN/, which shifts data into CH0FIFO. NEFUNC1 and NEFUNC0 tell the controller what kind of exchange is taking place as shown in Table 7.10. These determine when data should be shifted into the CH0 FIFO. The SCMODE2, SCMODE1, and SCMODE0 signals cause data to be shifted into the Voter. NEFUNC1, NEFUNC0, SCMODE2, SCMODE1, and SCMODE0 are initiated on the rising edge of MYFTC, and are used by the Synchronous Control only when MYBOUND is high.

NEFUNC1	NEFUNC0	Exchange Mode
0	0	Idle
0	1	Data XMIT
1	0	Data Reflect
1	1	SERP

Table 7.10 - Exchange Modes Indicated by NEFUNC

NEID2, NEID1, and NEID0 determine the NE's absolute location. SCMODE2, SCMODE1, and SCMODE0 determine which FIFO is to be shifted as shown in Table 7.11.

NE	SCMODE2	SCMODE1	SCMODE0
A	0	0	0
B	0	0	1
C	0	1	0
D	0	1	1
E	1	0	0
none	1	0	1
none	1	1	0
all	1	1	1

Table 7.11 - Reflecting FIFO Absolute Decoding

The SCMODE signals are converted to a five bit format (SRCA - SRCE) and then sent through an absolute to relative converter to become the CH0-CH4 signals. The IDT7202 FIFO outputs data on the high to low transition of a SHOUT/ signal, with an access time of 25 ns. However, on a low to high transition of a SHOUT/ signal, data is held valid for a maximum of 5 ns. Also data shifted out of the FIFOs must meet the setup and hold requirements of the pipe registers in the Voter chip (Chapter 8). To accomplish all of these requirements, we will make the SHOUT signal high for one-fourth of a clock cycle (20 ns) and low for three-fourths of a clock cycle (60 ns). This creates a data valid frame 40 ns wide. The 25% duty cycle of the SHOUT signal is created by using a D-flip flop, which uses a combination of the 12.5 MHz and 25 MHz signals, as shown in Figure 7.17.

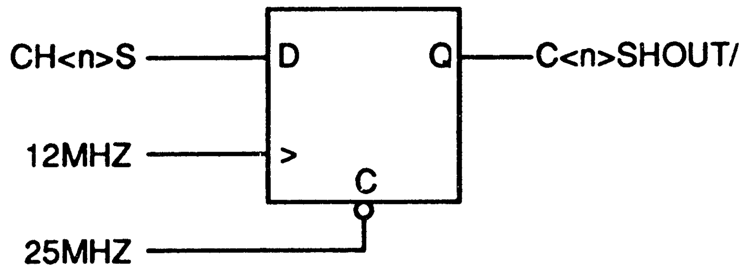


Figure 7.17 - SHOUT/ Generation

The equations for the Synchronous Control section are shown in Table 7.12. CH0SHIN/ is clocked at 25 MHz. Since voted data from the Voter chip is output one clock cycle (80 ns) after the data is shifted into the Voter chip, CH0SHIN/ is also delayed by two clock cycles.

SALL	= SCMODE2 • SCMODE1 • SCMODE0
SNONE	= (SCMODE2 • SCMODE1/ • SCMODE0)
	+ (SCMODE2 • SCMODE1 • SCMODE0/)
CHnS	= (CHn + SALL) • SNONE/
CH0SHIN	:= (NEFUNC0 + NEFUNC1) • MYBOUND
	delayed by one clock cycles

Table 7.12 - Synchronous Control Equations

7.7 VDAT Bus Driver

The VDAT Bus reads data from and transmits data to various sections of the NE. To assure that the correct data is driving the VDAT bus, the mechanism in Figure 7.18 is used. Data from the Debug Wrap Register, the Link Error Accumulator, or the Presence Accumulator is selected using multiplexers, depending on the Latch Decode signals.

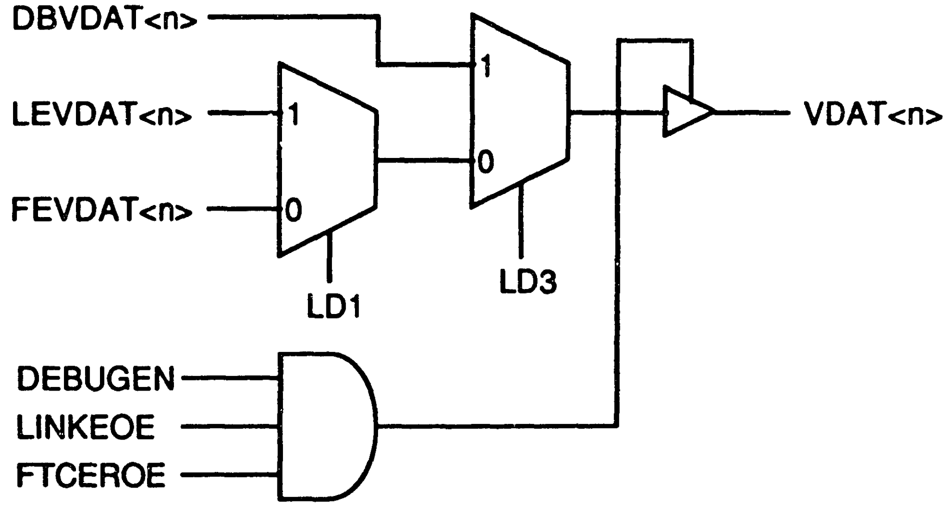


Figure 7.18 - VDAT Bus Driver Circuitry

Chapter 8
The Voter Chip

The Voter Chip can be divided into five functional blocks, the Latch Decode, the Mask Register, the Syndrome Accumulator, and the Voter sections. The block diagram for the Voter Chip is shown in Figure 8.1 The Latch Decode section is a decoder that generates signals for use throughout the Voter Chip. The Mask Register generates masks that are used by the Voter section. The Syndrome Accumulator records errors that occur during the voting process. The Voter is a five-way maskable, byte-wide, bit-for-bit majority voter. Voting errors are also generated by the Voter.

The Voter Chip is clocked at 12.5 MHz. For purposes of brevity 12.5 MHz is represented as 12MHZ on the schematic diagrams. The actual pinouts for the Voter Chip are found in Appendix F.

8.2 Latch Decode - Voter

The Latch Decode section in the Voter chip is identical to the one in the FTC Chip (see Section 7.3), except that different signals are decoded as shown in Table 8.1.

LD3	LD2	LD1	LD0	RESULTING OUTPUT
0	0	0	0	Idle (No output)
0	0	0	1	Resets the system (RESET/)
0	0	1	0	NEID Latch (NEIDLAT)
0	1	1	1	NE Mask Reg. Latch (NEMSKLAT)
1	0	0	0	PE Mask Reg. Latch (PEMSKLAT)
1	0	0	1	Syndrome Output Enable (SYNDEOE)
1	0	1	0	Voter Output Enable (VOTEOUT)

Table 8.1 - Voter Latch Decode Signals

RESET/ resets the Voter chip. NEIDLAT latches the VDAT2-0 signals to become the Network Element Identification (NEID) signals NEID2-0, respectively. The NEID signals indicate the absolute value of the NE as shown in Table 8.2.

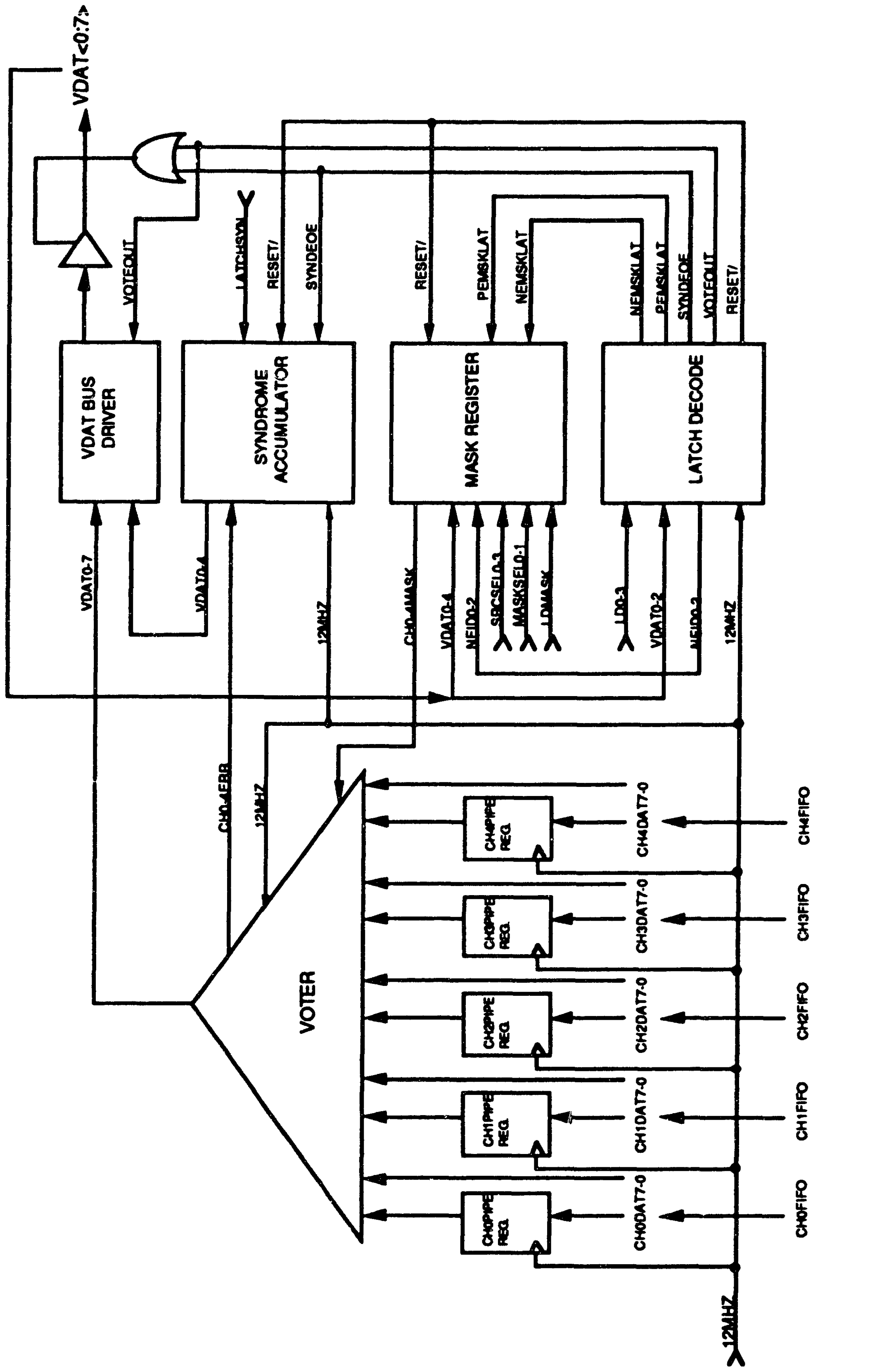


Figure 8.1 - Block Diagram of Voter Chip

NE	NEID2	NEID1	NEID0
A	0	0	0
B	0	0	1
C	0	1	0
D	0	1	1
E	1	0	0

Table 8.2 - NE Identification Signals

NEMSKLAT latches the VDAT4-0 signals as the Network Element masks and PEMSKLAT latches the VDAT4-0 signals as the Processor Element masks (see Section 8.2). SYNDEOE puts the Syndrome Accumulator errors onto the VDAT bus on the clock cycle after the Latch Decode signals are enabled and resets the syndrome errors (see Section 8.3). VOTEOUT enables voted data to be put on the VDAT bus on the clock cycle after the Latch Decode signals are enabled (see Section 8.4). The Latch Decode is a basic decoder built out of combinational logic and registers. RESET/ is registered. All of the other signals are purely combinational signals.

8.3 Mask Register

The Mask Register section provides masks for the voter. It has three types of masks, the Reflect Masks, the Vote Masks, and the From Masks. Each group of masks consists of five bits (one for each Fault Containment Region) and are active high. The masks are selected by the MASKSEL inputs as shown in Table 8.3.

MASKSEL1	MASKSEL0	MASK GENERATED
0	0	Reflect Masks (Source Masks/)
0	1	Vote Masks (NE Masks • PE Masks)
1	-	From Masks (NE Masks • Source Masks)

Table 8.3 - Mask Register MASKSEL Outputs

Masks that are selected by MASKSEL1-0 are latched on to a register to become the CH0-4MASK signals used by the voter when LDMASK is enabled. The overall block diagram for the Mask Register section is shown in Figure 8.2.

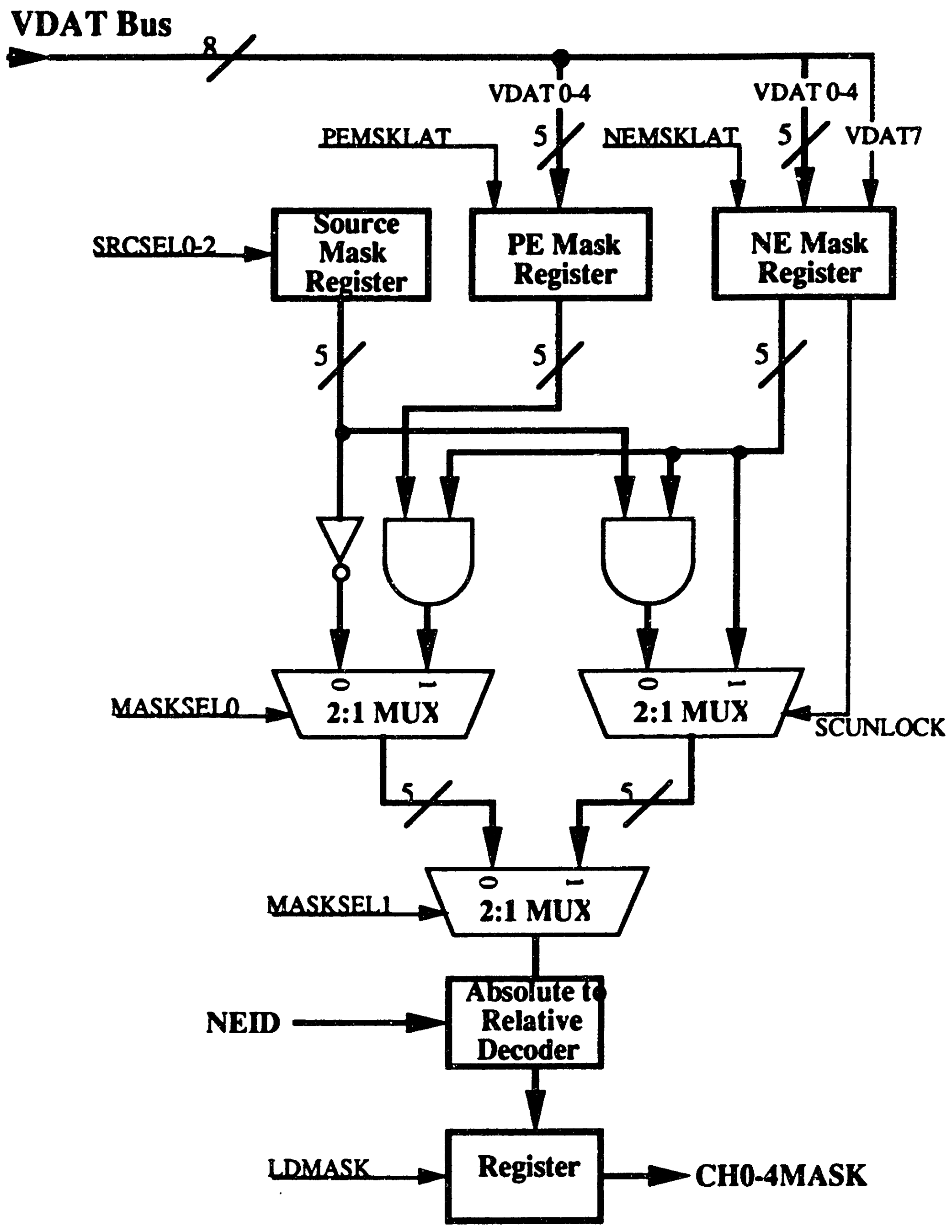


Figure 8.2 - Block Diagram of Mask Register

The Reflect Masks are used during the first round of Class Two exchanges and mask out the NEs that are not reflecting data. They are simply the inverted values of the Source Masks. The Vote Masks are used during Class One exchanges and mask out NEs and PEs that have been determined to be faulty. They are the NE Masks ANDed with the PE Masks. The From Masks are used during SERP operations and during the second round of Class Two exchanges. They reflect the data from each non-faulty NE. They are the NE Masks ANDed with the Source Masks, except in triplex operation where they are simply the NE Masks.

The three types of masks are generated from three latched sets of masks, the NE Masks, the PE Masks, and the Source Masks. The the NE Masks are latched in absolute format from the low five bits of the VDAT bus by NEMSKLAT. The high VDAT7 bit is used to latch the SCUNLOCK signal which is used during triplex mode of operation. This selects the proper type of From Mask. The VDAT bits are latched as shown in Table 8.4.

VDATBIT	NEMASKLATCH
7	SCUNLOCK
5-6	Unused
4	NEEMASK
3	NEDMASK
2	NECMASK
1	NEBMASK
0	NEAMASK

Table 8.4 - VDAT NE Mask Assignments

NE Masks are normally latched during Configuration Table updates. The mechanism used for latching the NE Masks into the NE Mask Register is a series of combinational logic and synchronous latches. This mechanism is diagrammed in Figure 8.3. SCUNLOCK is latched in the same manner.

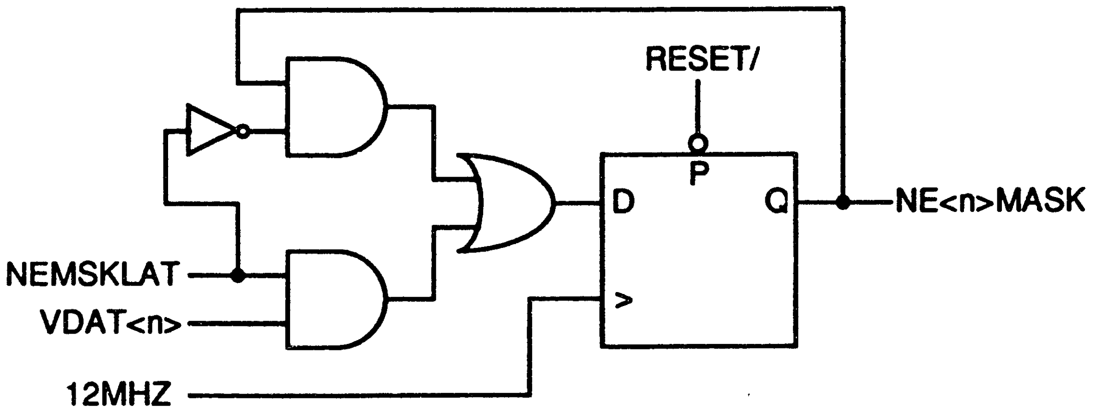


Figure 8.3 - NE Mask Register Circuitry

The PE Masks are used in Class One exchanges as Vote Masks to mask out a faulty processor of a given Fault Masking Group (FMG), and need to be updated before each Class One data exchange if the previous exchange was not with the same FMG, or the processor failure information has changed. The PE Masks are latched in absolute format from the low five bits of the VDAT bus by PEMSKLAT. The VDAT bits are latched as shown in Table 8.5.

VDATBIT	PEMASKLATCH
5-7	Unused
4	PEEMASK
3	PEDMASK
2	PECMASK
1	PEBMASK
0	PEAMASK

Table 8.5 - VDAT PE Mask Assignments

The mechanism used for latching the PE masks into the PE Mask Register is a series of combinational logic and synchronous latches. This mechanism is diagrammed in Figure 8.4.

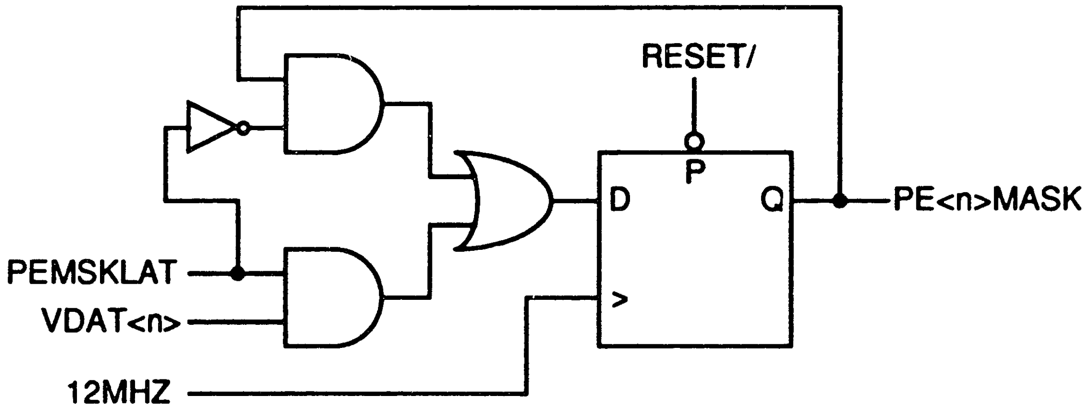


Figure 8.4 - PE Mask Register Circuitry

Three control signals, SRCSEL0, SRCSEL1, and SRCSEL2 are taken straight from the Global Controller to indicate what type of Source Mask should be used. When Class Two exchange is occurring, an NE reflects data to the other NEs, using the Reflect Masks, which are generated from the Source Masks. When a SERP is occurring, non-faulty elements vote on SERP data, using the From Masks, which are generated by the Source Masks and NE Masks. The Source Mask generation is shown in Table 8.6.

SRCSEL2	SRCSEL1	SRCSEL0	Exchange Type	SC<E-A>MASK
0	0	0	From A or SERP Reflect A	11110
0	0	1	From B or SERP Reflect B	11101
0	1	0	From C or SERP Reflect C	11011
0	1	1	From D or SERP Reflect D	10111
1	0	0	From E or SERP Reflect E	01111

Table 8.6 - Exchange Types and Mask Generation

The masks generated by the SRCSEL signals are implemented using combinational logic and latches.

Finally, the masks are converted from absolute to relative format using the absolute to relative converter shown in Figure 8.5.

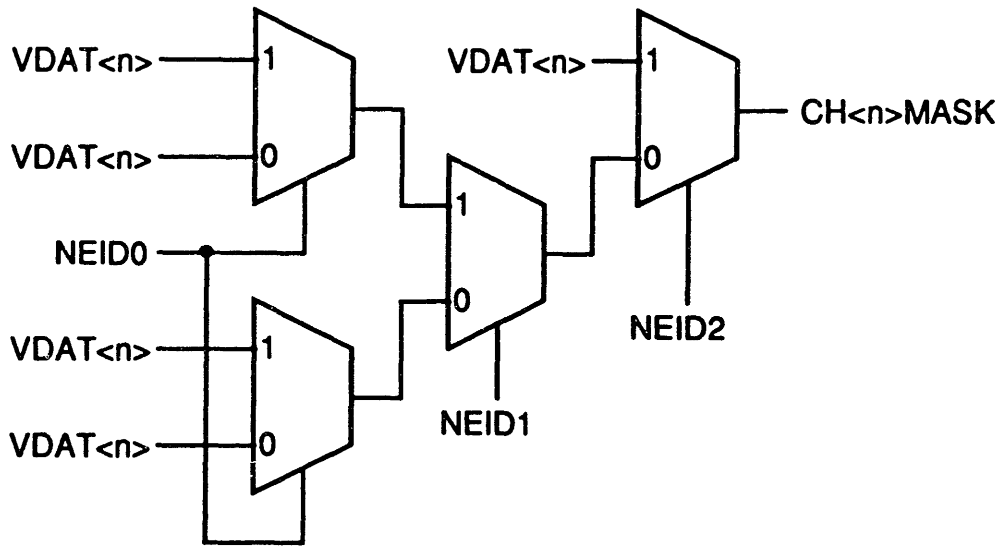


Figure 8.5 - Absolute to Relative Converter

8.4 Syndrome Accumulator

The Syndrome Accumulator records errors that are detected by the voter. Errors are recorded as a high signal only when LATCHSYN is enabled. This information is requested from the Global Controller when needed by the SYNDEOE signal. The error bits are written to the VDAT bus in relative form as shown in Table 8.7. These signals are taken directly from the voter.

VDATBIT	SYNDROMEERROR
5-7	Set Low
4	CH4 Error
3	CH3 Error
2	CH2 Error
1	CH1 Error
0	CH0 Error

Table 8.7 - Syndrome Accumulator Error Format

An error is set high when a voted data byte differs from the respective NE's data byte. The Syndrome Accumulator errors are placed on the VDAT bus by the mechanism shown in Figure 8.6. The errors are output on the VDAT bus when the SYNDEOE signal is sent to the VDAT Bus Driver.

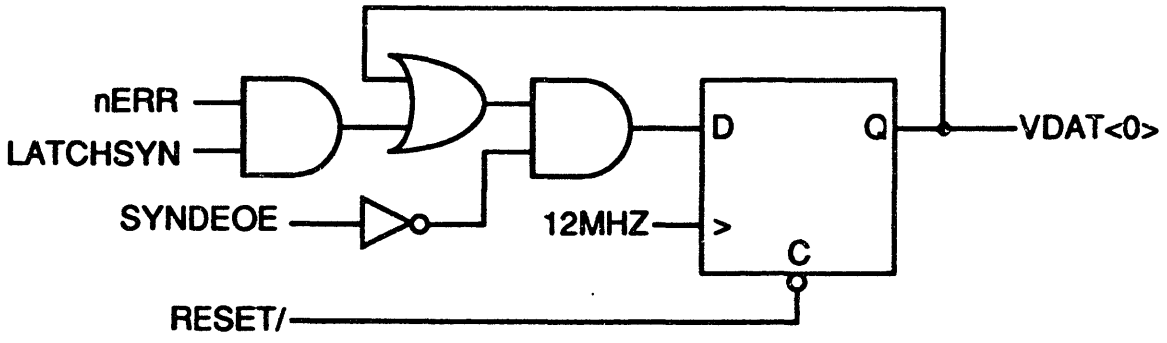


Figure 8.6 - Latching Syndrome Errors on the VDAT Bus

8.5 Voter

The Voter is basically a five channel, byte wide, maskable, bit-for-bit majority voter. The Voter uses the mask signals from the Mask Register, CH0MASK, CH1MASK, CH2MASK, CH3MASK, and CH4MASK, to mask out NEs or PEs that are determined to be faulty or are not a part of the sending Fault Masking Group (FMG). It generates CH0ERR, CH1ERR, CH2ERR, CH3ERR, and CH4ERR which indicate an error in that relative NE's data. The data being voted is temporarily held in pipeline registers. The overall block diagram for the Voter circuit is shown in Figure 8.7, where the gray area represents that the voter error logic is registered.

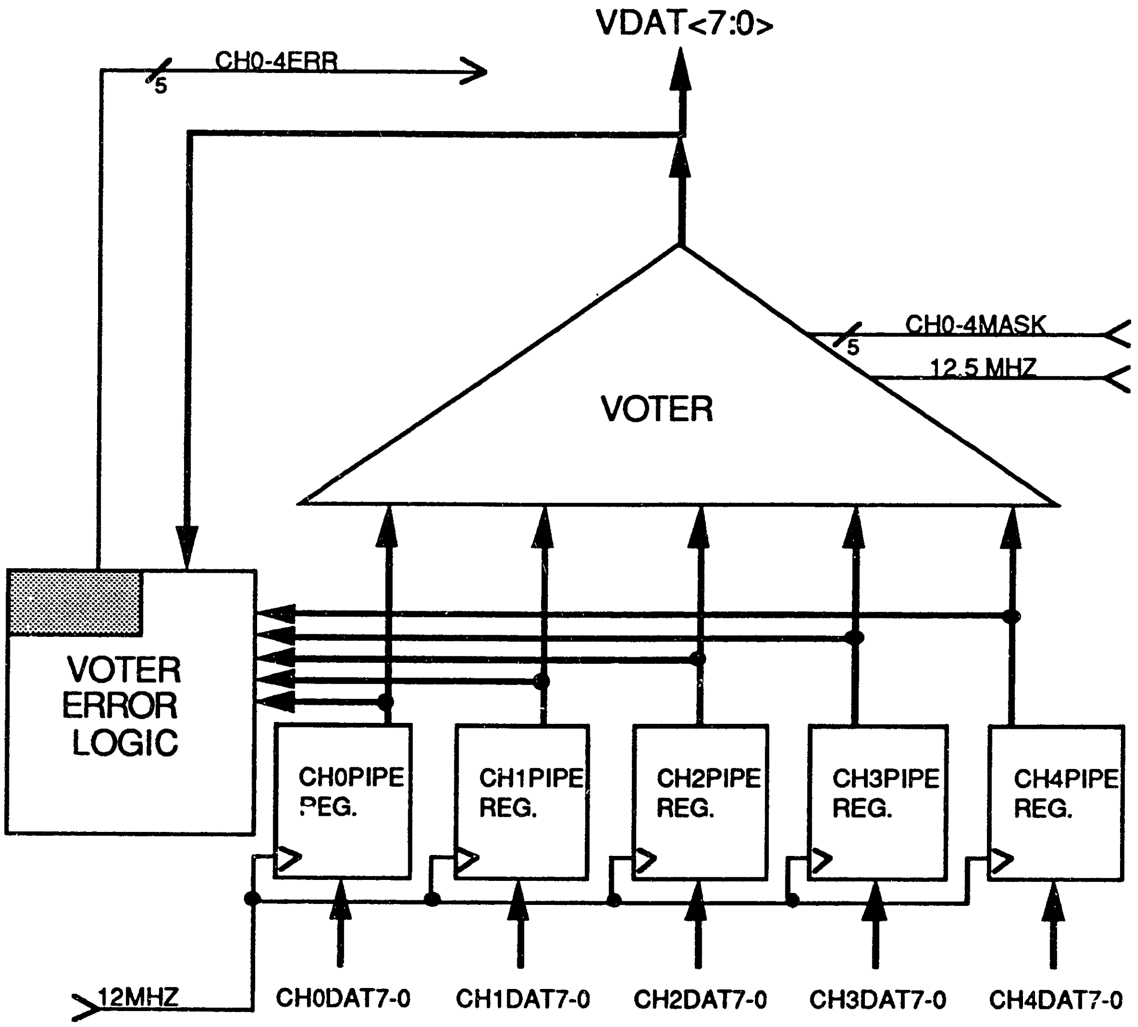


Figure 8.7 - Voter Block Diagram

The pipeline registers are simply latches clocked by the 12.5 MHz clock signal as shown in Figure 8.8.

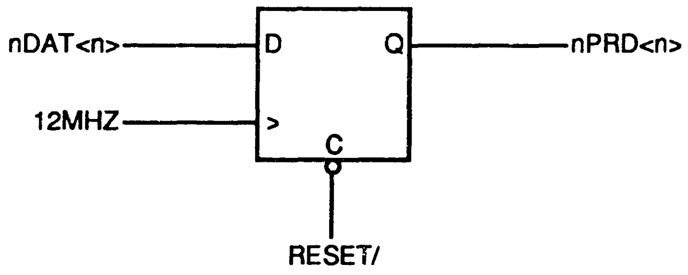


Figure 8.8 - Voter Pipeline Register

Implementing the five channel, byte wide, maskable, bit-for-bit majority voter, is not a trivial task. The output involves combinations of eighty variables, which are five data bytes and five mask bytes, yielding 2^{80} ($\approx$1.2E24) possible combinations. Pure combinational logic, when fully reduced can yield thirty-five product terms per bit, for a total of 280 product terms. However, this process is simplified since we are not considering quintuplex voting. For non-quintuplex voting, we simply set the output to high if one of the two following cases is true:

1) two or more valid data bits are high, or
2) simplex voting is used and the valid data bit is high.

A valid data bit means that the bit is not masked out by the Mask Register signals. The truth table for non-quintuplex voting is shown in Table 8.8, where M<0:4> represent the mask bits, D<0:4> represent the data bits, and - means that the input is irrelevant to the output. Voted output is done using expander terms for speed purposes.

M4	M3	M2	M1	M0	D4	D3	D2	D1	D0	Output
-	-	-	1	1	-	-	-	1	1	1
-	-	1	-	1	-	-	1	-	1	1
-	1	-	-	1	-	1	-	-	1	1
1	-	-	-	1	1	-	-	-	1	1
-	-	1	1	-	-	-	1	1	-	1
-	1	-	1	-	-	1	-	1	-	1
1	-	-	1	-	1	-	-	1	-	1
-	1	1	-	-	-	1	1	-	-	1
1	-	1	-	-	1	-	1	-	-	1
1	1	-	-	-	1	1	-	-	-	1
0	0	0	0	1	-	-	-	-	1	1
0	0	0	1	0	-	-	-	1	-	1
0	0	1	0	0	-	-	1	-	-	1
0	1	0	0	0	-	1	-	-	-	1
1	0	0	0	0	1	-	-	-	-	1

Table 8.8 - Truth Table for Non-Quintuplex Voting

Errors are generated when the output of a given channel differs from the voted output on VDAT<7:0>. These errors can be detected by using XOR gates as shown in Figure 8.9. In the actual schematics, the error detection logic uses combinational terms using macrocells with three product terms for speed purposes and since the voter logic needs most of the expander terms.

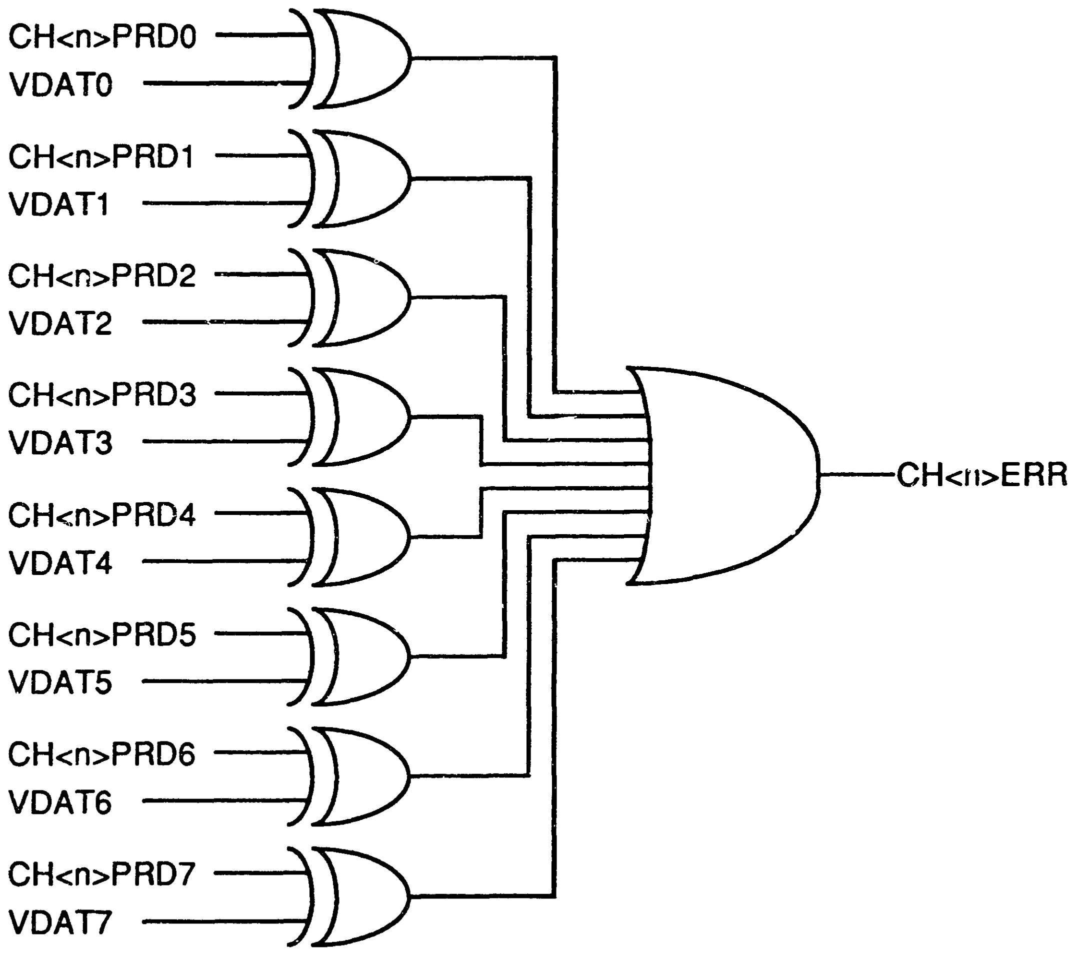


Figure 8.9 - Voter Error Detection Logic

8.5 VDAT Bus Driver

The VDAT bus is used for the transmission of data throughout the local NE. In order to accomplish this, the VDAT bus must be bidirectional and must be able to be driven by the correct data. This is accomplished by the mechanism in Figure 8.10.

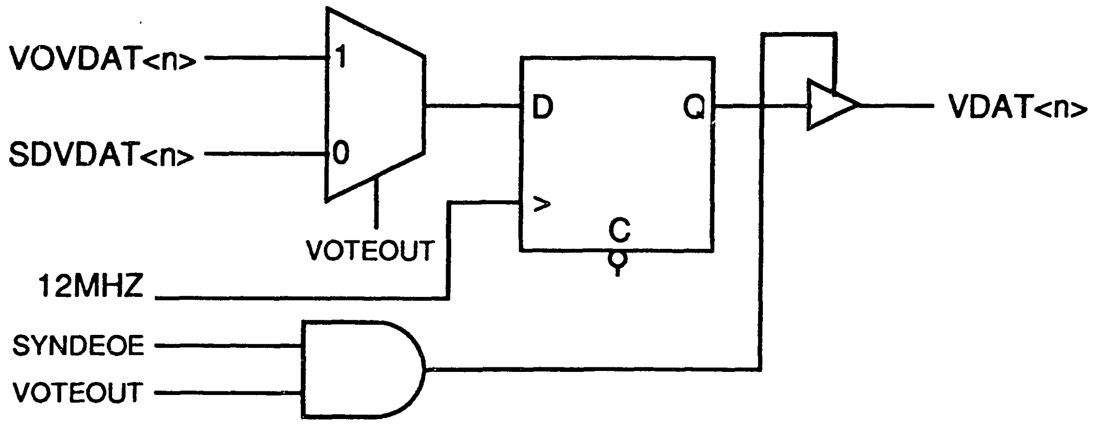


Figure 8.10 - VDAT Bus Driver Logic

Chapter 9
Simulation and Testing

Simulation for the FTC FPGA and Voter FPGA were done using Altera's Maxplus system. This system allows use of a waveform editor whereby any portion of the design can be simulated by generating input waveforms and observing the outputs. The FTC and Voter FPGAs could not actually be tested with the rest of the C3 FTPP since this is the first completed portion of the C3 FTPP. However, by using timing parameters from other portions of the C3 FTPP, a timing analysis can be done to test functionality.

Simulation was done in a bottom-up fashion, where small portions of the design were simulated first, then larger blocks, and finally entire chips. This was done to assure functionality of the individual blocks before larger tests. Signals from the NE Global Controller arrive have maximum propagation delay of 12 ns with respect to the rising edge of the 12.5 MHz clock. Data read from the FIFOs arrives 25 ns after a strobe falling edge and is held for a least 5 ns after the rising edge of the strobe. When writing to the FIFOs, there is a setup time of 15 ns and a hold time of 0 ns relative to a strobe rising edge. Using these external timing parameters and the simulation results of inter-chip signals, the functionality of the Data Path Chip Set may be determined.

9.1 Timing Model

Calculating the timing for the EPM5128 chips is a straightforward task since the EPM5128 has fixed propagation delays depending on the internal layout of each chip. To calculate propagation delays of a signal, we can use a timing model given in the Altera databook. This timing model is shown in Figure 9.1.

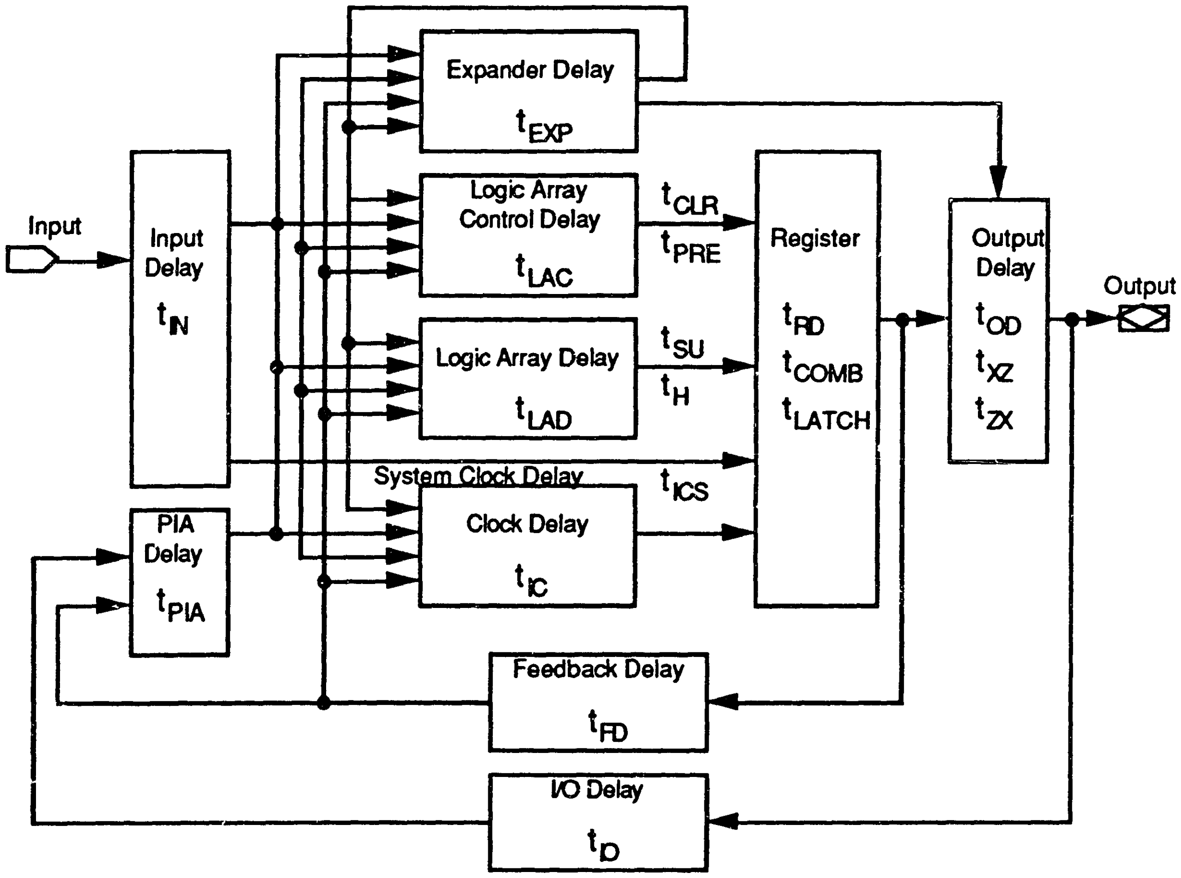


Figure 9.1 - EPM5128 Timing Model

By knowing the timing parameters for each chip, we can calculate the propagation delays for any output given the timing of the inputs. The timing parameters for the EPM5128-1 (the fast version of the EPM5128) are shown in Table 9.1.

External Timing Parameters

Symbol	Parameter	Min	Max
t_{PD1}	Input to non-registered output		25
t_{PD2}	I/O Input to non-reg. output		40
t_{SU}	Setup time	15	
t_{H}	Hold time	0	
t_{CO1}	Clock to output delay		14
t_{ASU}	Asynchronous setup time	5	
t_{AH}	Asynchronous hold time	6	
t_{CH}	Clock high time	8	
t_{CL}	Clock low time	8	
t_{ACO1}	Async. clock to output delay		25
t_{CNT}	Minimum clock period		20

Internal Timing Parameters

Symbol	Parameter	Min	Max
t_{IN}	Input pad and buffer delay		5
t_{IO}	I/O Input pad and buffer delay		6
t_{EXP}	Expander array delay		12
t_{LAD}	Logic array delay		12
t_{LAC}	Logic control array delay		10
t_{OD}	Output buffer and Pad delay		5
t_{ZX}	Output buffer enable delay		10
t_{XZ}	Output buffer disable delay		10
t_{SU}	Register setup time	6	
t_{LATCH}	Flow-through latch delay		3
t_{RD}	Register delay		1
t_{COMB}	Combinatorial delay		3
t_{H}	Register hold time	6	
t_{IC}	Clock delay		14
t_{ICS}	System clock delay		2
t_{FD}	Feedback delay		1
t_{PRE}	Register preset time		5
t_{CLR}	Register clear time		5
t_{PIA}	Progr. Interconn. Array delay		14

Table 9.1 - EPM5128 Propagation Delays

The timing parameters and model are used by the Maxplus simulator so it is not necessary to know these values. However, it is helpful when determining how the chip circuitry needs to be laid out in order to meet speed requirements.

9.2 The Fault Tolerant Clock Chip

For purposes of simplicity, the simulation of the Fault Tolerant Clock chip will be broken up into seven sections, one for each part of the chip itself. Finally the simulation of the entire chip will be discussed. For all parts of the chip, data coming from the VDAT bus must satisfy a setup time of 5 ns and a hold time of 6 ns.

9.2.1 Asynchronous Control

The Asynchronous Control generates signals that shift data into the FIFOs. A data strobe for the correct channel should be generated when one of three cases happen: when the channel's data strobe goes high, when the channel's control strobe and violation strobe both go high, and when a channel's debug mode is enabled. The results of simulating the Asynchronous Control block for the first case can be seen in Figure 9.2. The SHIN/ signal has a maximum propagation delay of 40 ns from the rising edge of the DSTRB signal.

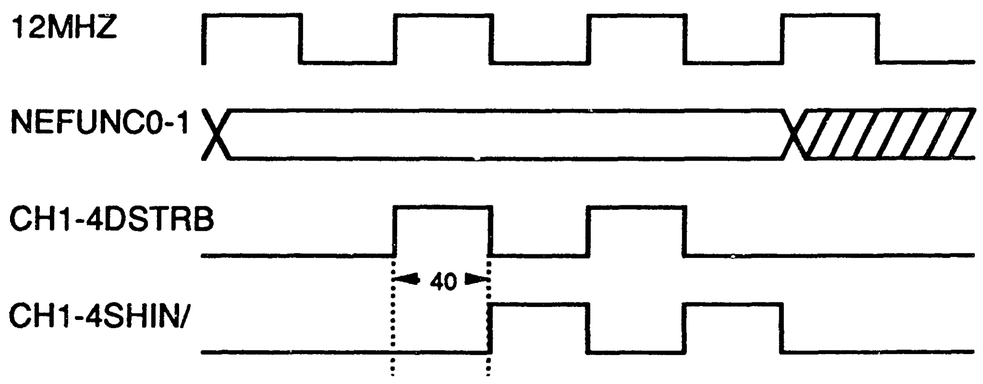


Figure 9.2 - Asynchronous Control Simulation for Normal Operation

The simulation results for the second case are identical to those of the first case, except that the control strobe and the violation both go high instead of the data strobe. In either of these two modes, the SHIN/ signals meet the setup and hold times for the FIFO with respect to valid data from the TAXI chip. Simulating the Asynchronous Control block in debug mode yields a signal that is identical to the results of CH0SHIN/ from the Synchronous Control. The results can be seen in Figure 9.3. This is done since data needs to be shifted into the FIFOs synchronously when in debug mode.

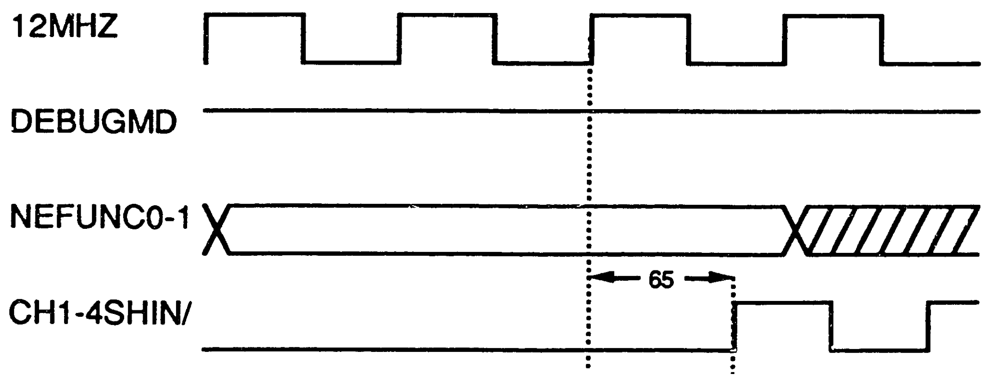


Figure 9.2 - Asynchronous Control Simulation in Debug Mode

In debug mode, the SHIN/ signals meet the setup and hold times for the FIFO with respect to valid data from the Voter Chip.

9.2.2 Debug Wrap Register

The Debug Wrap Register should latch a byte of data (call it X) from the VDAT bus when the DEBUGLT signal is strobed (generated by the Latch Decode input of 1100). Data is output onto the VDAT bus when the DEBUGOE signal is strobed (generated by the Latch Decode input of 1101). The simulation results for the Debug Wrap Register are shown in Figure 9.4.

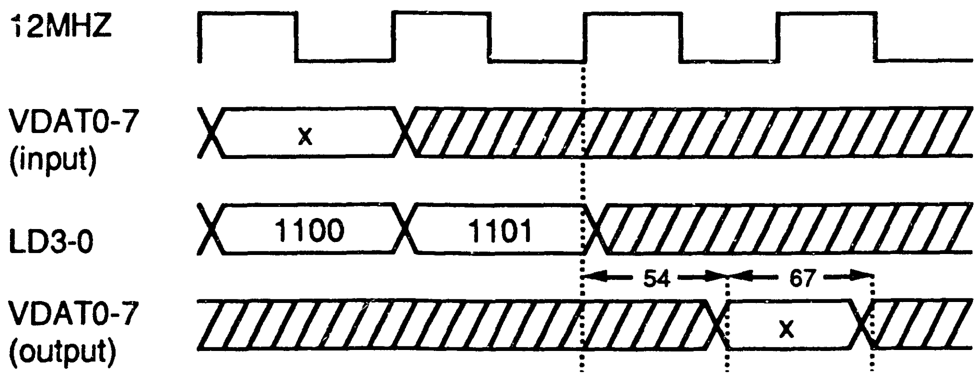


Figure 9.4 - Debug Wrap Register Simulation

9.2.3 Fault Tolerant Clock

The simulation of the Fault Tolerant Clock includes many smaller simulations. The Bane State Machine determines whether the local NE's FTC is synchronized with the rest of the NEs' FTCs. It does this by generating three signals, ERRWIN, SAW, and SBW. By comparing these signals with the NE's BOUND signals, we can determine whether the local FTC is ahead, behind, or in error. ERRWIN, which stands for Error Window, is a signal that drops low for three and a half clock cycles (280 ns), representing when a channel's BOUND signal is not in error. SA is a window that indicates when a local FTC is ahead of the other FTCs. SBW is a window that indicates when a local FTC is behind the other FTCs. By symmetry, the error window encloses one clock cycle where SAW is high, one clock cycle where SBW is high, and one and a half clock cycles where neither SAW or SBW are high. These signals are sent to the Synchronization Feedback section and then to the My State Machine so that the local FTC will be adjusted with respect to the other FTCs. Note that the Bane State Ma^hine is clocked at 25 MHz and these error windows are adjusted by the propagation delay. The simulation results of the Bane State Machine are shown in Figures 9.5 and 9.6.

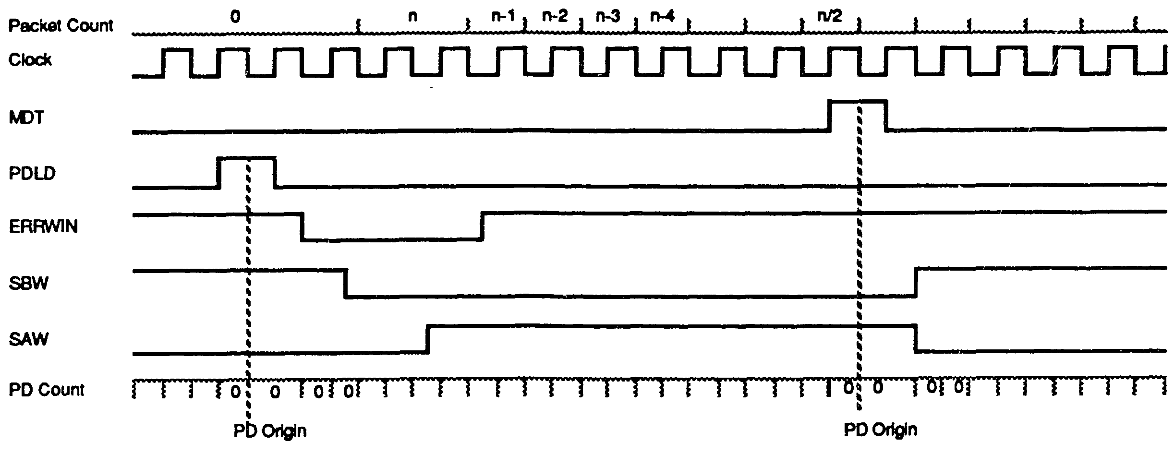


Figure 9.5 - Bane State Machine Simulation for No Propagation Delay

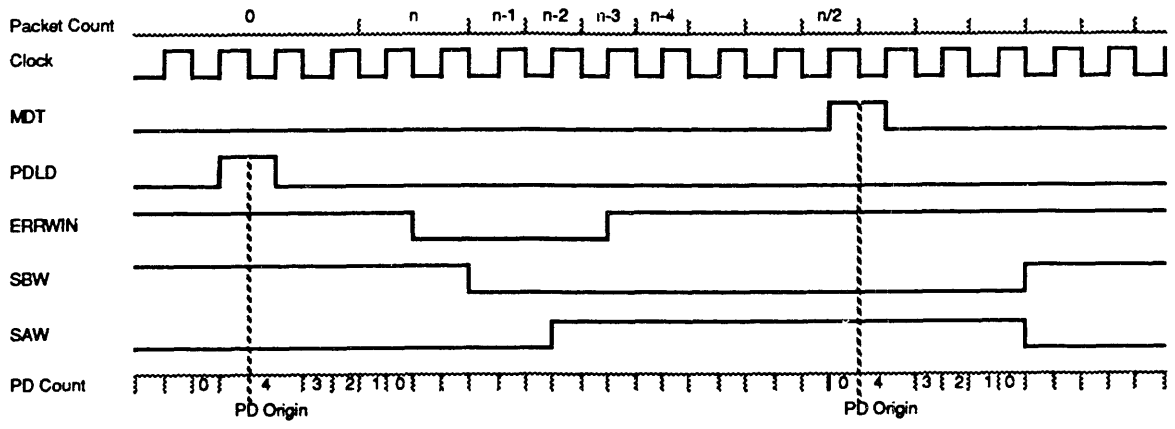


Figure 9.6 - Bane State Machine Simulation for a Propagation Delay of Four

The Bound Generator generates BOUND signals for channels 1 to 4. A BOUND signal is a signal that is high during the period that data is being strobed in, and low otherwise. Due to the nature of the fiberoptic receivers and transmitters, when data is not being strobed in, control strobes are always being sent (CSTRB). This signal can be used to mark the end of a channel's BOUND signal. However, control strobes are also sent when violations occur (VLTN). This should not mark the end of a BOUND signal since and error could occur in the middle of a transmission. The Bound Generator also generates the MEDBOUND signal, which stands for the Median Bound. MEDBOUND is simply the second BOUND signal. The results of the simulation of the Bound Generator are shown in

Figure 9.7. MEDBOUND is shown for the case that all the CH1-4B signals are the same. Otherwise, MEDBOUND follows the second CH1-4B signal by 40 ns.

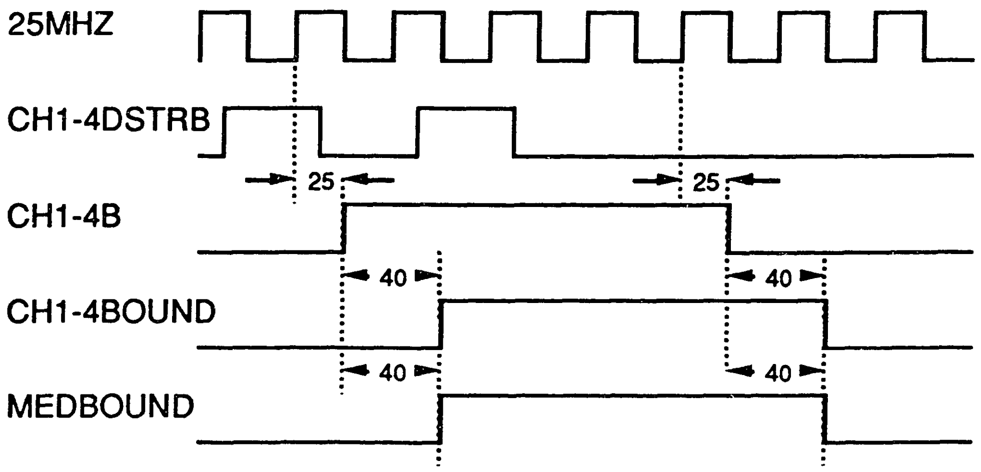


Figure 9.7 - Bound Generator Simulation

The Mask Latch should simply latch masks from the VDAT4-0 signals when NEMSKLAT is strobed. These masks are used by the Bound Generator just described. Figure 9.8 shows the simulation of the FTC Mask Latch.

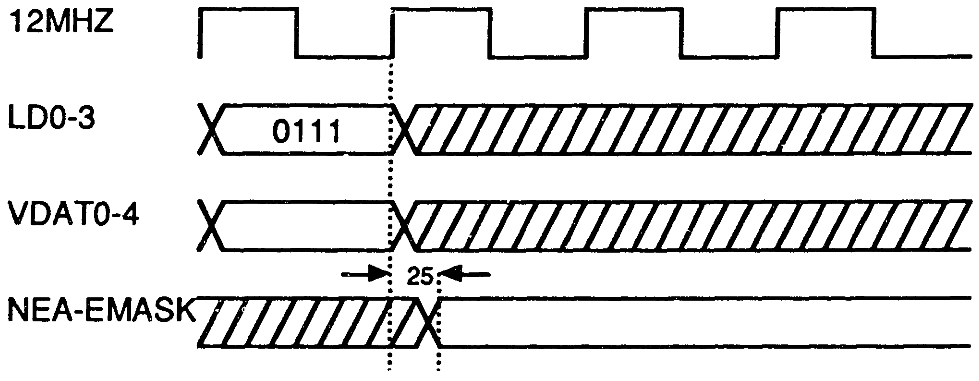


Figure 9.8 - FTC Mask Latch Simulation

The Mid Data Transfer should strobe MDT/ when the packet count is equal to half of the latched packet count. The Mid Data Transfer simulation is shown in Figure 9.9.

This signal is used by the Propagation Delay Counter to reload the propagation delay and by the Bane State Machine.

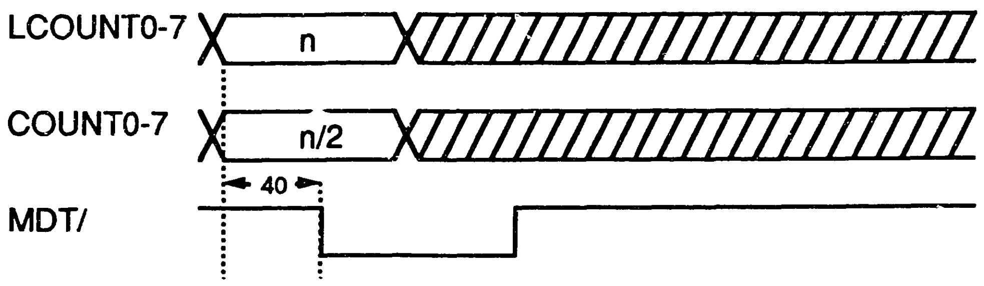


Figure 9.9 - Mid Data Transfer Simulation

The My State Machine generates the MYBOUND, MYFTC, PKTLD/, and PDLD/ signals and should skip State C when the system is behind, and should go through and extra state (State D) when the system is ahead. The My State Machine is the synchronism mechanism that adjusts the local NE's clock with respect to the other clocks. The results are shown in Figure 9.10, 9.11, and 9.12.

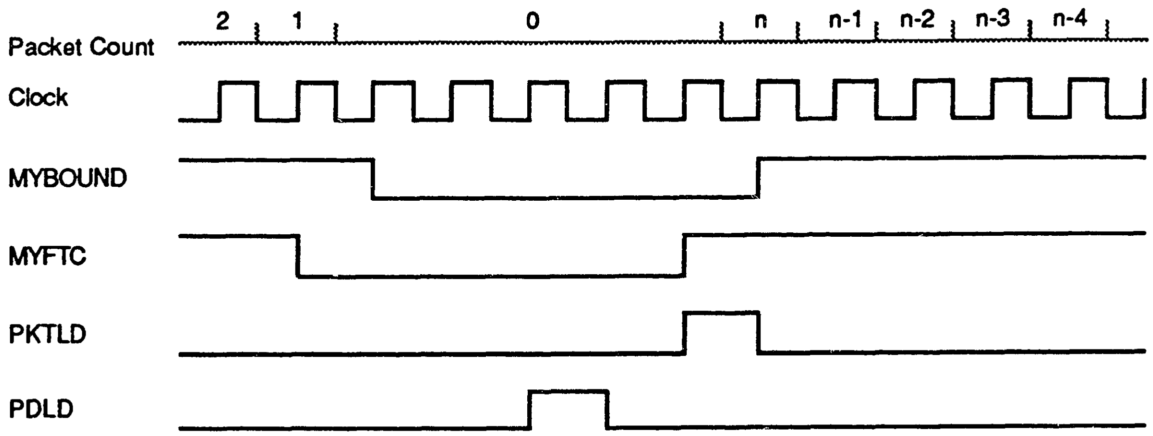


Figure 9.10 - FTC My State Machine Simulation (Ahead)

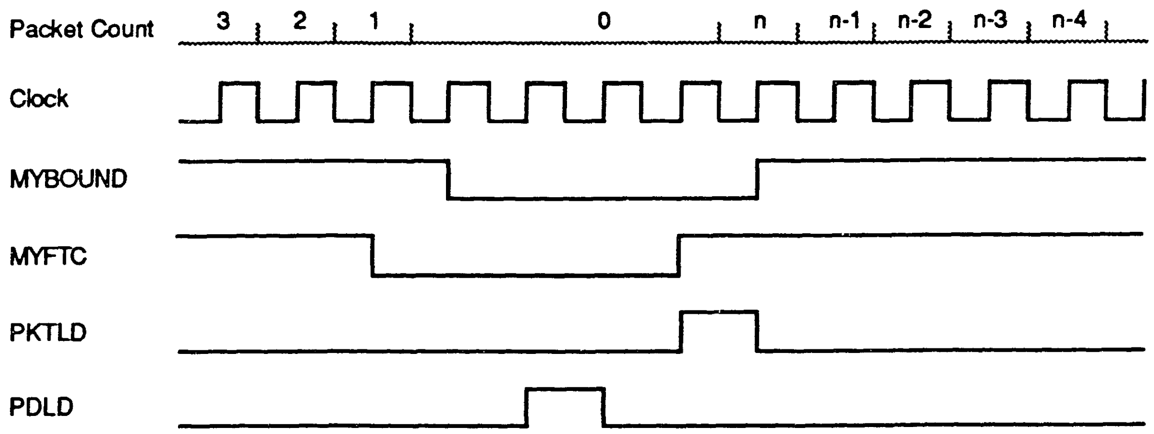


Figure 9.11 - FTC My State Machine Simulation (Normal)

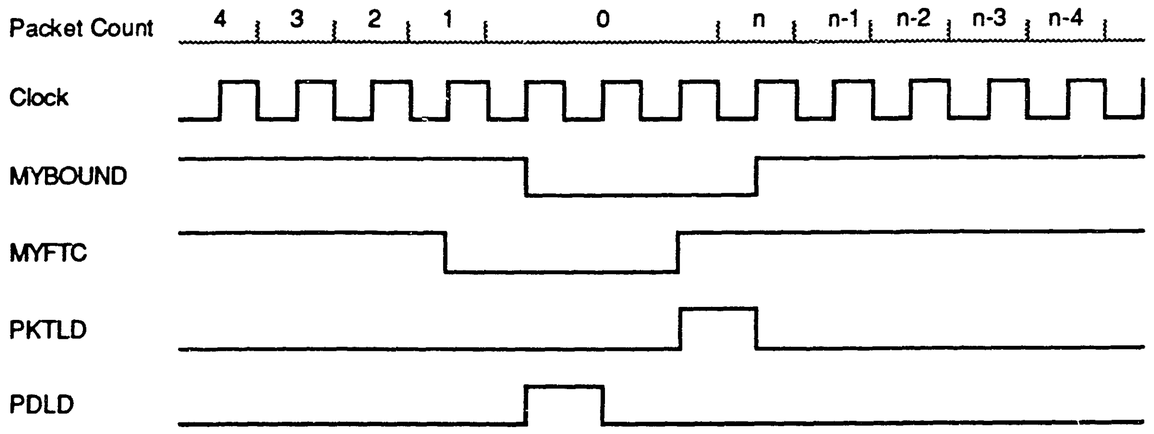


Figure 9.12 - FTC My State Machine Simulation (Behind)

The Packet Latch and Counter latches VDAT7-0 onto a counter when FTCPL is strobed (generated by the Latch Decode input of 0011). It should reset the counter to this value when PKTLD/, which comes from the My State Machine is strobed. The results of the Packet Latch and Counter simulation are shown in Figure 9.13.

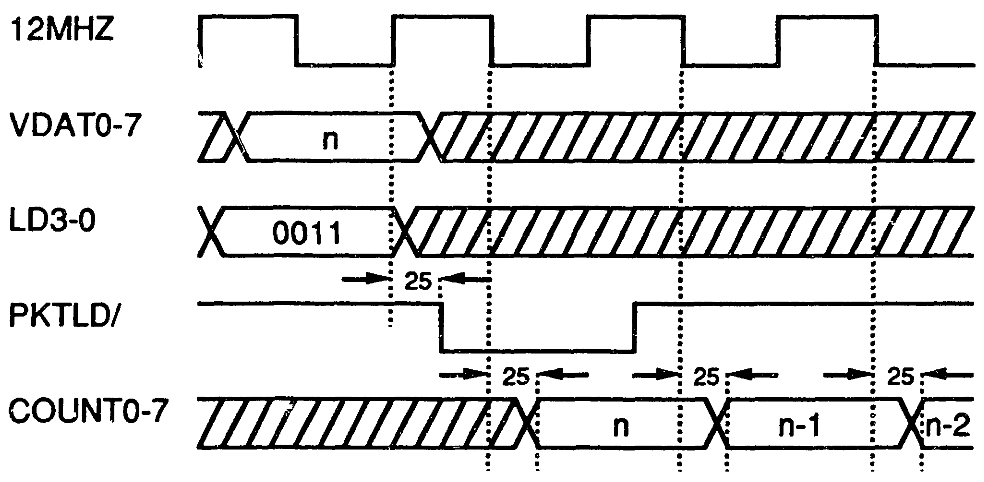


Figure 9.13 - FTC Packet Latch and Counter Simulation

The Presence Accumulator does two things. First, it takes a sample of each BOUND signal from the Bound Generator at 25 MHz. Secondly, it compares this sampled signal to the BOUND signal to find the rising edge of each BOUND signal. This rising edge is compared to ERRWIN and errors are recorded if the BOUND signal rises outside of the error window. The results of the simulation of the Presence Accumulator are shown in Figure 9.14.

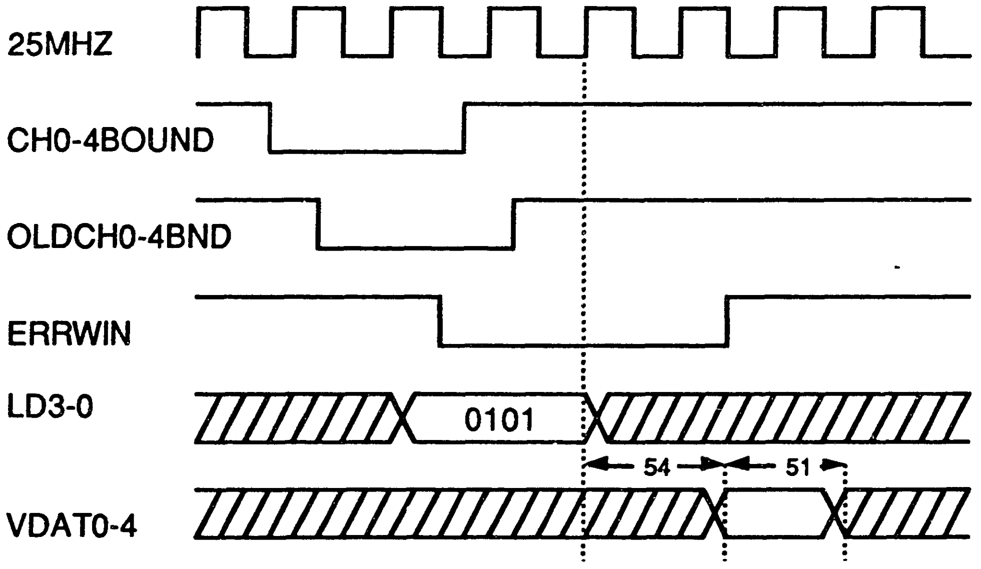


Figure 9.14 - Presence Accumulator Simulation

The Propagation Delay Latch and Counter should latch VDAT5-0 onto a counter when FTCDL is strobed (generated by the Latch Decode input of 0100). It should reset the counter to this value when PDLD/ from My State Machine or MDT/ from the Mid Data Transfer section is strobed. It then counts down on each rising edge of the 25 MHz clock and halts when it reaches zero. The results of the Propagation Delay Latch and Counter simulation are shown in Figure 9.15.

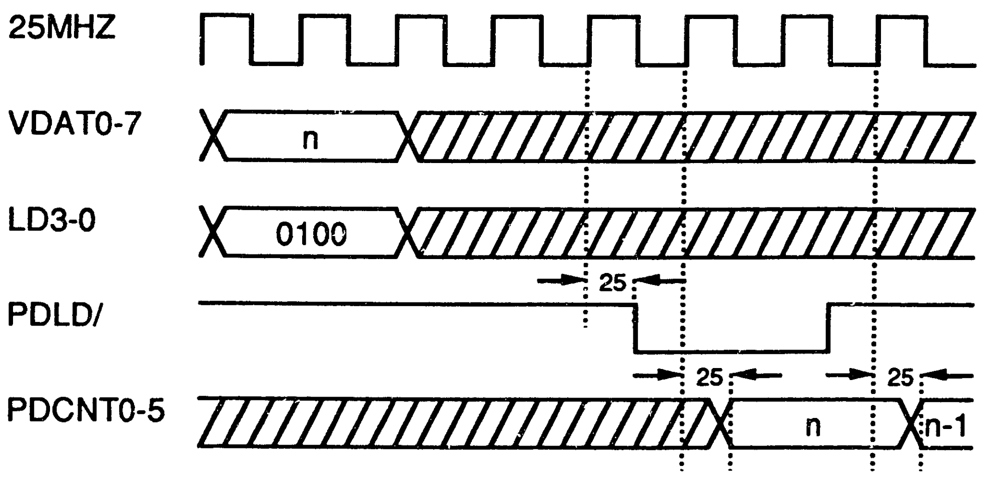


Figure 9.15 - FTC Propagation Delay Latch and Counter Simulation

9.2.4 Latch Decode - FTC

The Latch Decode signal generates internal signals depending on the LD3-0 inputs. These signals must satisfy a setup time of 5 ns and a hold time of 6 ns with respect to the 12.5 MHz clock. All the signals are combinational except for RESET/, which is registered. The NE identification signals are also latched by the Latch Decode section. The results of the simulation of the FTC Latch Decode are shown in Figure 9.16.

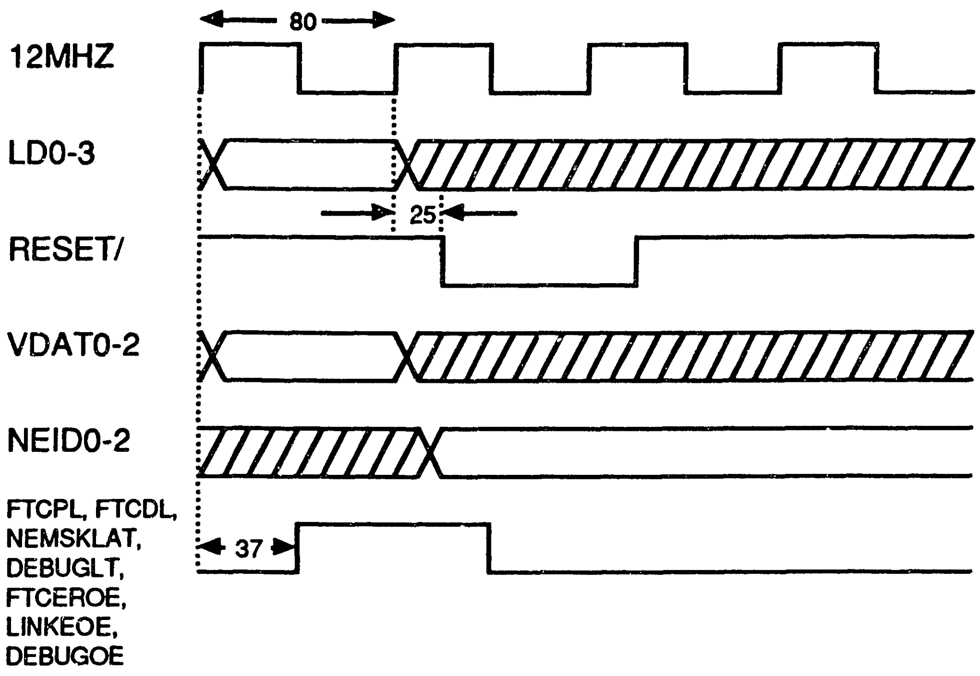


Figure 9.16 - FTC Latch Decode Simulation

9.2.5 Link Error Accumulator

The Link Error Accumulator records errors that occur when data is received through the fiberoptic receivers. When an error occurs, the VLTN signal from the respective fiberoptic receiver is strobed. These are recorded by the Link Error Accumulator until either RESET/ or LINKEOE is strobed (generated by the Latch Decode inputs of 0001 or 0110, respectively). LINKEOE puts the link error results onto the VDAT bus. The results of simulating the Link Error Accumulator are shown in Figure 9.17.

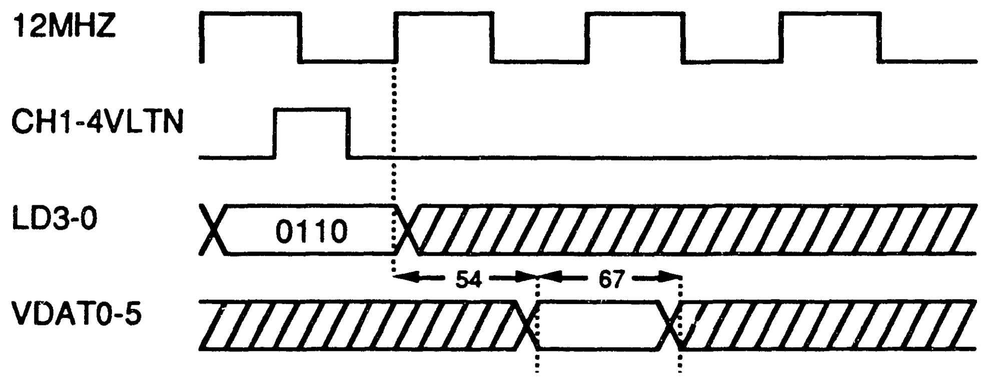


Figure 9.17 - Link Error Accumulator Simulation

9.2.6 Synchronous Control

The Synchronous Controller generates signals that synchronously shift out data from FIFOs (CH1-4SHOUT) and the signal that shifts in data into the local NE's FIFO (CH0SHIN/). These signals are generated depending on the SRCSEL0-2 signals and the function of the NE (NEFUNC0-1). The results of the simulation of the Synchronous Control is shown in Figure 9.18.

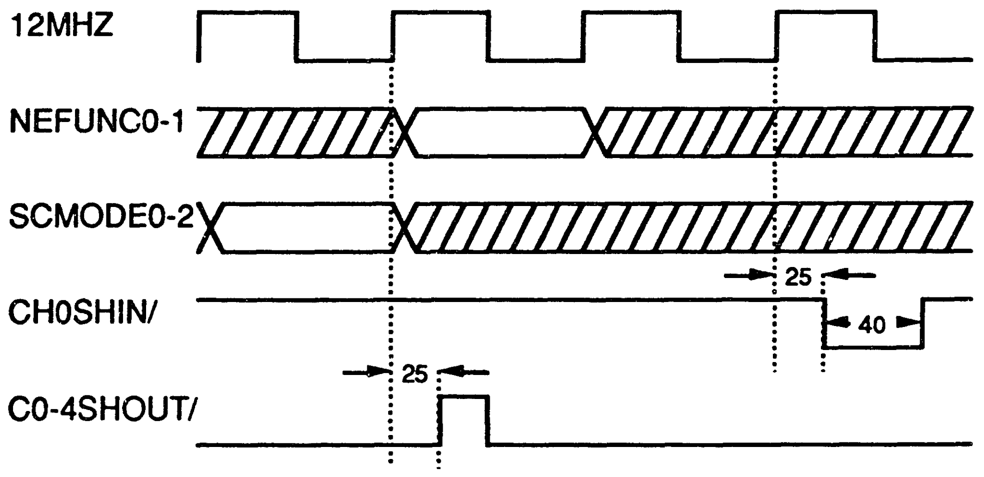


Figure 9.18 - Synchronous Control Simulation

Data from a FIFO can be read 25 ns after the falling edge of SHOUT and is held valid until 5 ns after the rising edge of SHOUT.

9.2.7 VDAT Bus Driver

The VDAT Bus Driver selects data from certain sections of the FTC chip. It selects data from the Debug Wrap Register, the Link Error Accumulator, or the Presence Accumulator, depending on the Latch Decode signals. The simulation of the VDAT Bus Driver is shown in Figure 9.19.

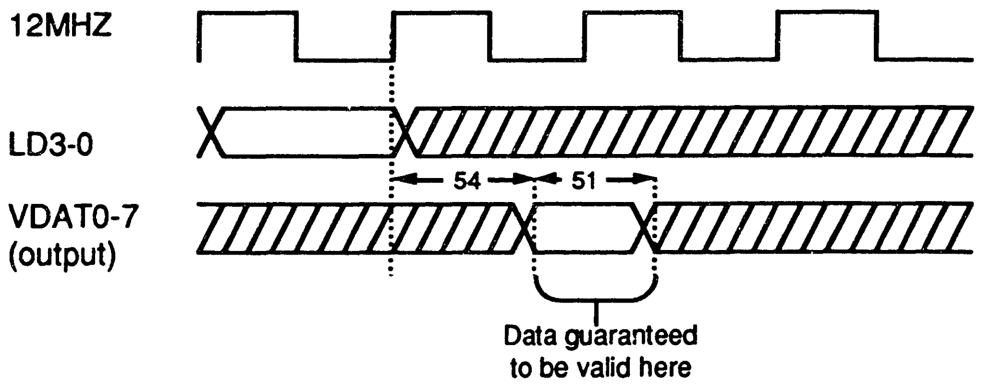


Figure 9.19 - FTC VDAT Bus Driver Simulation

9.2.8 Overall FTC Simulation

The simulation of the entire FTC chip is difficult to depict. This is because there are so many variables that depicting every different case would be infeasible. The FTC Chip was simulated using several different modes, with errors injected at certain points. The FTC synchronization mechanism worked as expected. To get an idea of how the FTC Chip works, see Appendix C, which shows a simulation of what signals would look like for Network Element A in a normal mode of operation with a propagation delay of six and a packet length of twelve. During actual use of the C3 FTPP, the packet length will probably be sixty-four for normal message exchanges. Since there are normally four cycles in between message exchanges which are used for synchronization, 94% of the bandwidth of the fiberoptic receivers and transmitters may be used when using that packet length.

Although simulation can only be done one chip at a time, we can approximate how the chip will function in the presence of other NEs by estimating input waveforms. After thorough simulation, it is expected that the FTC Chip will synchronize itself with respect to the other NEs by means of message exchanges. Additional simulation beyond the capabilities of the Altera simulator is needed to conclusively prove that synchronization will be achieved under all circumstances.

9.3 The Voter Chip

The Voter Chip handles the majority voting of data and the reflecting of input data for proper modes of exchange. The Voter Chip simulation will be broken up into five sections, one for each part of the Voter Chip. Finally, the simulation of the entire chip will be discussed. For all parts of the chip, data coming from the VDAT bus or from the FIFOs must satisfy a setup time of 5 ns and a hold time of 6 ns.

9.3.1 Latch Decode - Voter

The Latch Decode signal generates internal signals depending on the LD3-0 inputs. These signals must satisfy a setup time of 5 ns and a hold time of 6 ns. All of the signals are combinational, except for RESET/, which is registered. The results of the simulation of the Voter Latch Decode are shown in Figure 9.20.

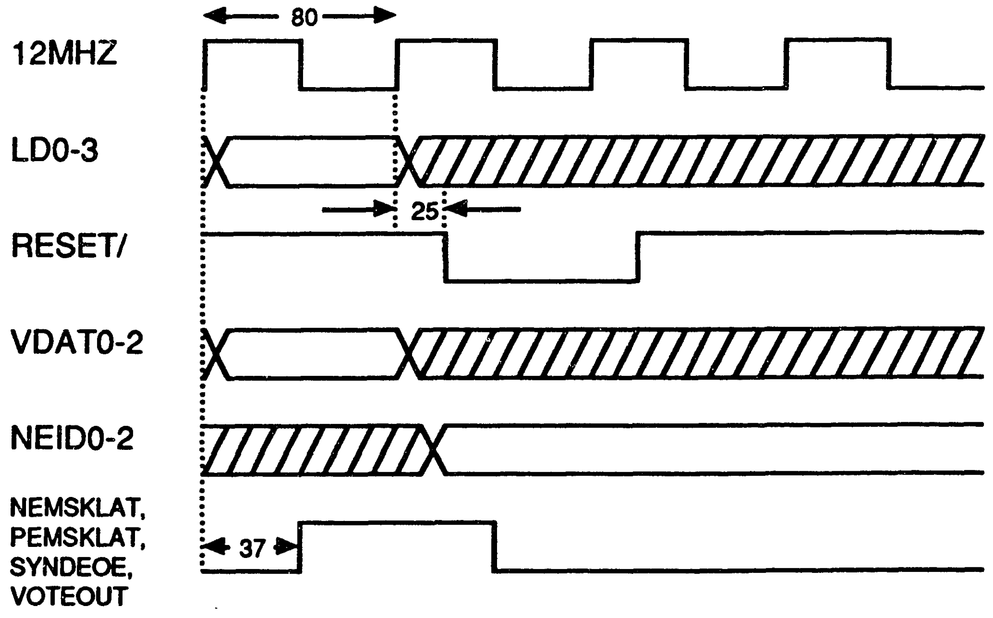


Figure 9.20 - Voter Latch Decode Simulation

9.3.2 The Mask Register

The Mask Register selects masks depending on operation of the NE. The Mask Register includes three mask latches, the NE mask latch, the PE mask latch, and the Source Mask Latch. The NE masks and the PE masks are latched from the VDAT bus by the NEMSKLAT and PEMSKLAT signals from the Latch Decode section. The Source masks are latched from the SRCSEL0-2 signals. These masks or combinations of these masks are selected depending on the MASKSEL0-1 signals. The selected masks become active only when LDMASK is enabled. The results of the simulation of the Mask Register is shown in Figure 9.21.

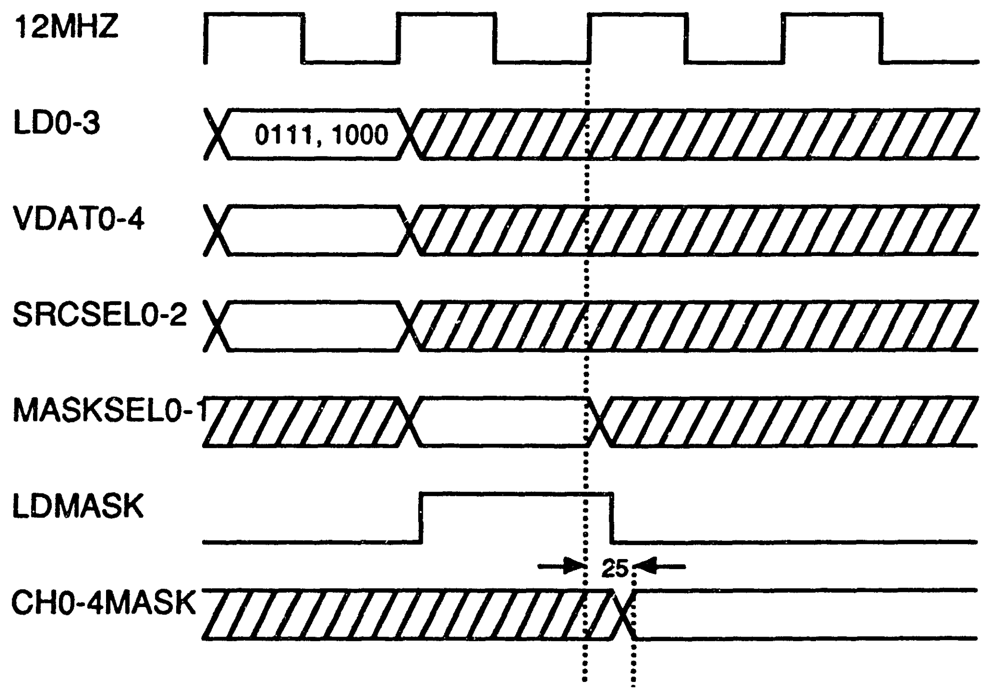


Figure 9.21 - Mask Register Simulation

9.3.3 Syndrome Accumulator

The Syndrome Accumulator records errors that occur from the Voter. The error results are reset when either SYNDEOE or RESET/ (generated by the Latch Decode inputs of 1001 or 0001, respectively) are strobed. SYNDEOE enables the error results to be output onto the VDAT bus. The simulation of the Syndrome Accumulator is shown in Figure 9.22.

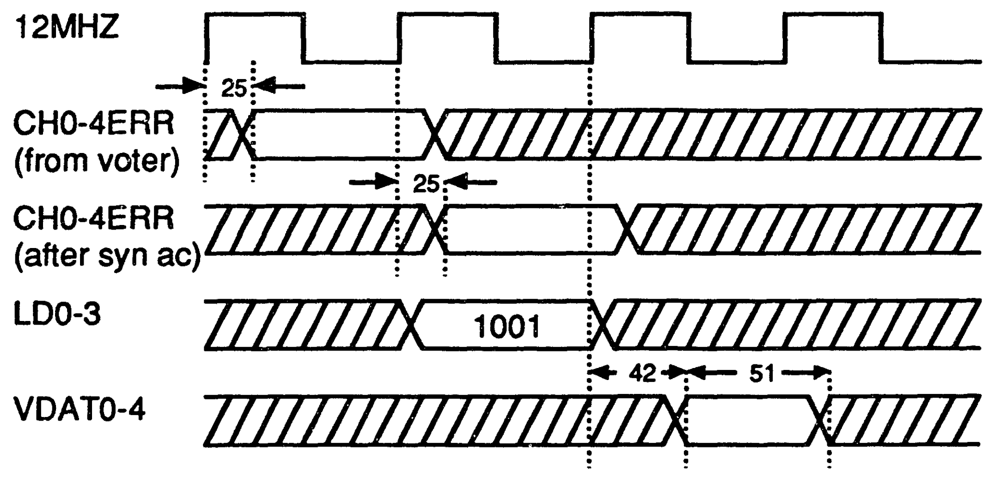


Figure 9.22 - Syndrome Accumulator Simulation

9.3.4 Voter

The Voter section majority votes each bit of data from five channels. It also generates errors when a channel has data that disagrees with the voted data. The results of the simulation of the Voter are shown in Figure 9.23.

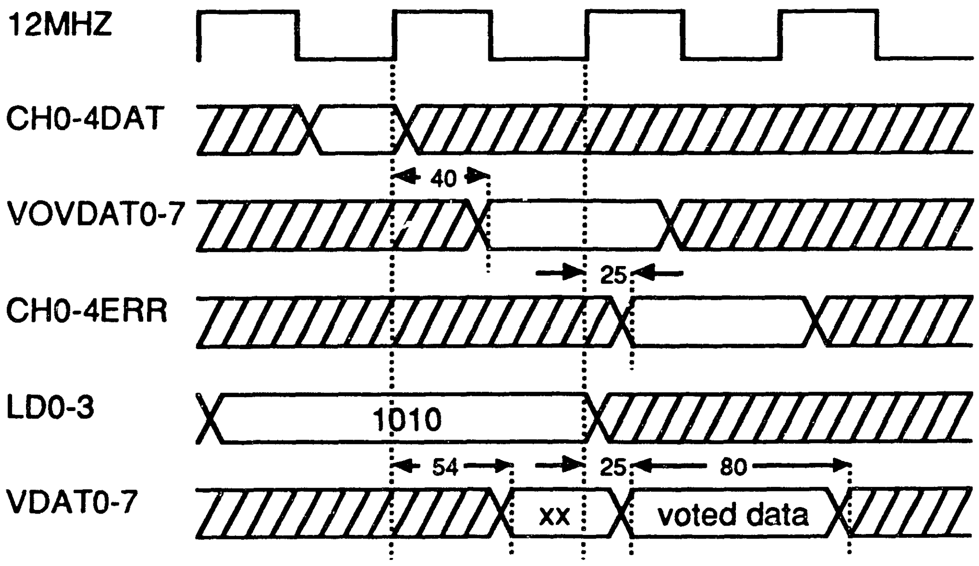


Figure 9.23 - Voter Simulation

The output of the voter meets the setup time and hold time of the FIFOs with respect to the CH0SHIN/ signal.

9.3.5 VDAT Bus Driver

The VDAT Bus Driver selects data from certain sections of the Voter chip. It selects data from the Voter or the Syndrome Accumulator, depending on the Latch Decode signals. The simulation of the VDAT Bus Driver is shown in Figure 9.24.

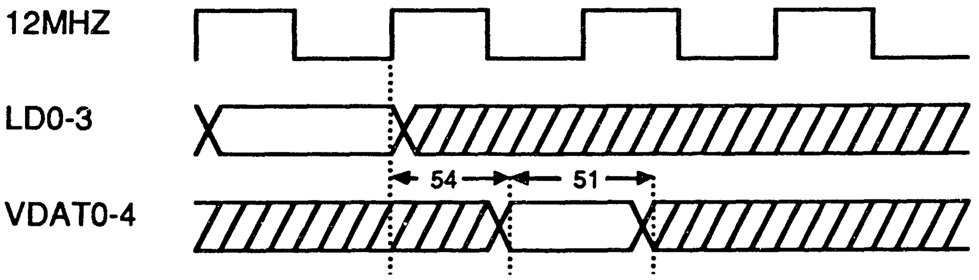


Figure 9.24 - FTC VDAT Bus Driver Simulation

9.3.6 Overall Voter Simulation

The Voter Chip takes signals from the Global Controller to assure that data is properly voted. As with the FTC Chip, enumerating the simulation cases would be infeasible, so only a few operational modes will be shown. There are three basic modes that the Voter chip functions in, one for each type of mask. Three sample vote operations are shown in Appendix D. After thorough testing of the Voter Chip, it has been determined that it properly majority votes data from five maskable channels.

9.4 FTC to Voter Interface

The FTC Chip and Voter Chip are related in that they need to be synchronized so data read from FIFOs (controlled by the FTC Chip) is valid for the Voter Chip, and data written to the FIFOs from the Voter Chip is strobed into the FIFOs properly by means of the FTC chip. The FTC Chip generates synchronous SHOUT/ signals that shift data out of the FIFOs. The data shifted out of the FIFOs must meet the 5 ns setup time of the Voter Chip. The Voter Chip outputs data that is clocked into the chip one cycle later with a delay. The FTC Chip generates a signal (CH0SHIN/) that synchronously shifts in data into CH0FIFO at the appropriate time, meeting the 15 ns setup time of the FIFO. When in debug mode the FTC shifts data into all the FIFOs from the Voter Chip. The interface between the FTC and Voter Chips are shown in Figure 9.25.

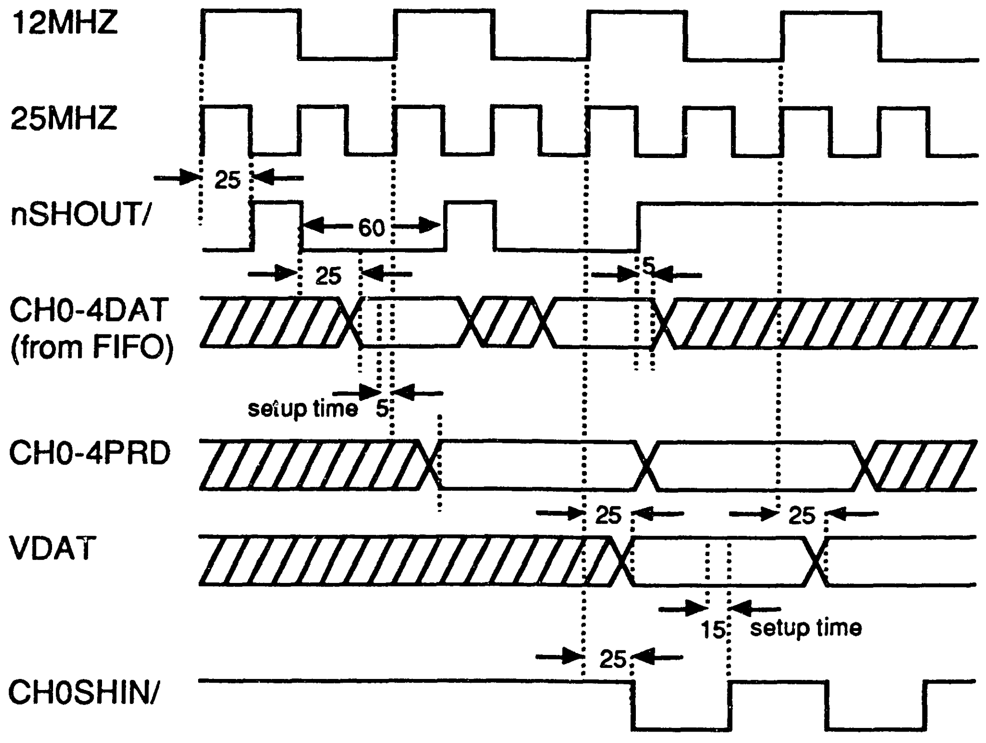


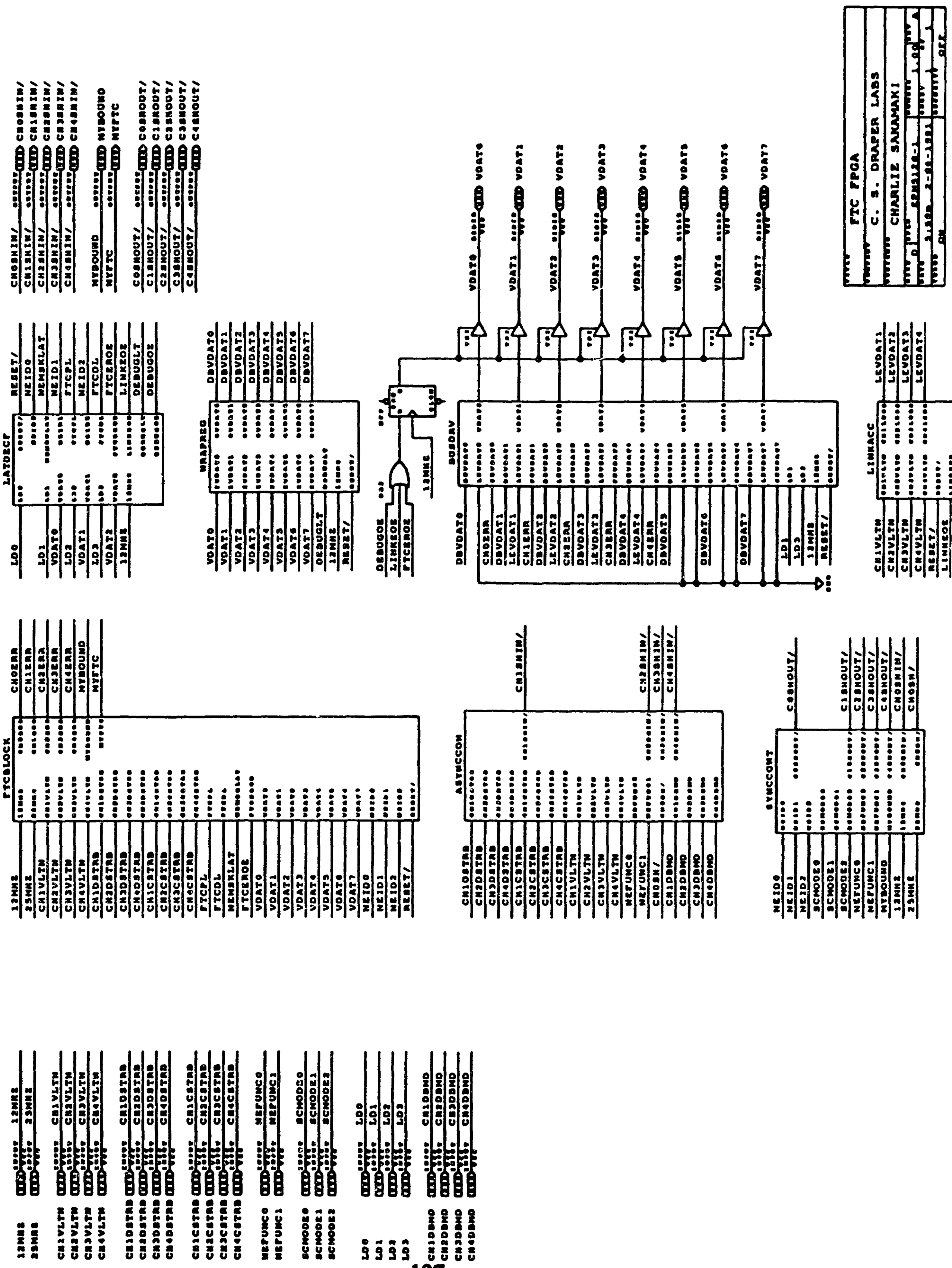
Figure 9.25 - FTC to Voter Interface Timing

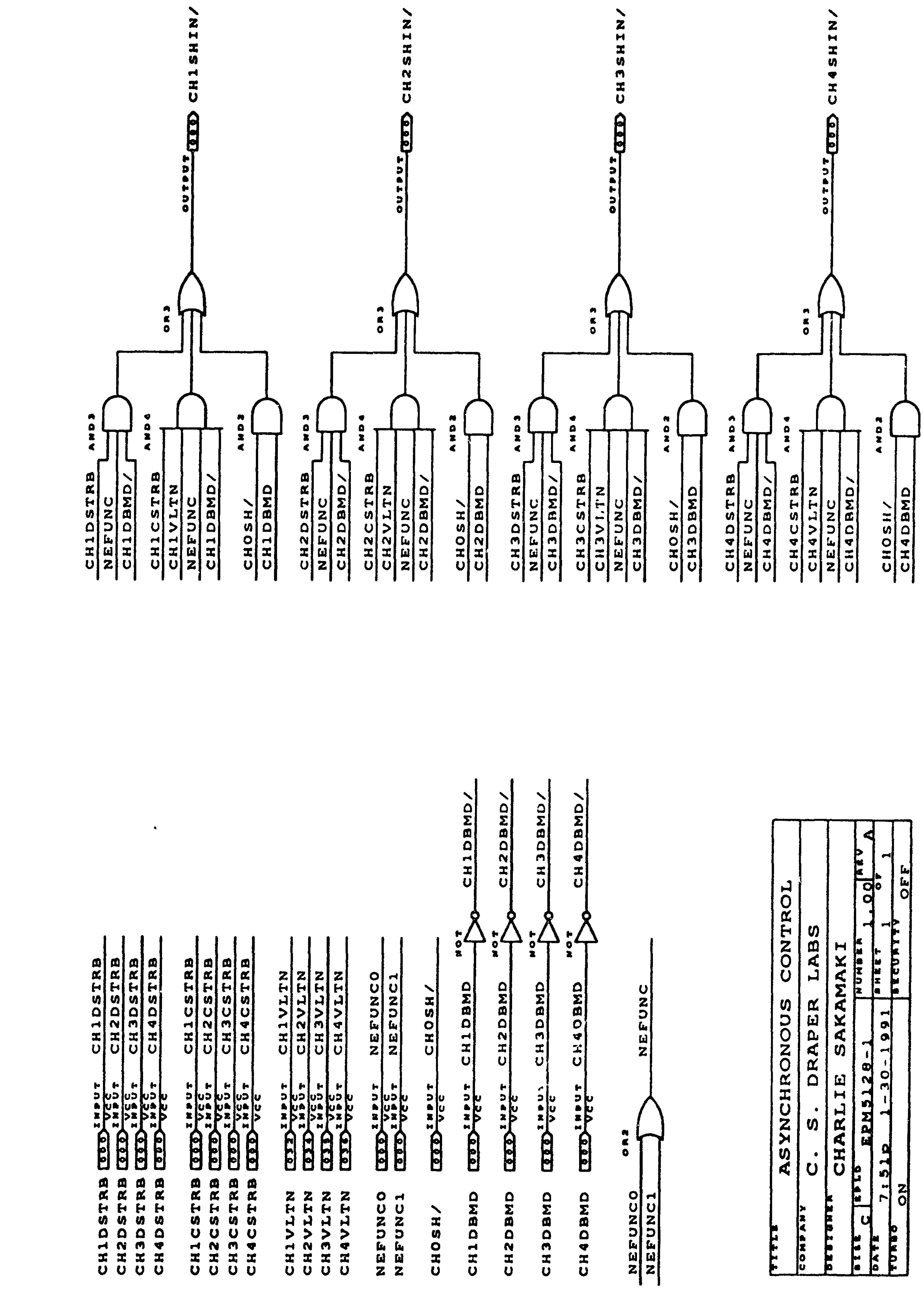
Chapter 10
Conclusions and Recommendations

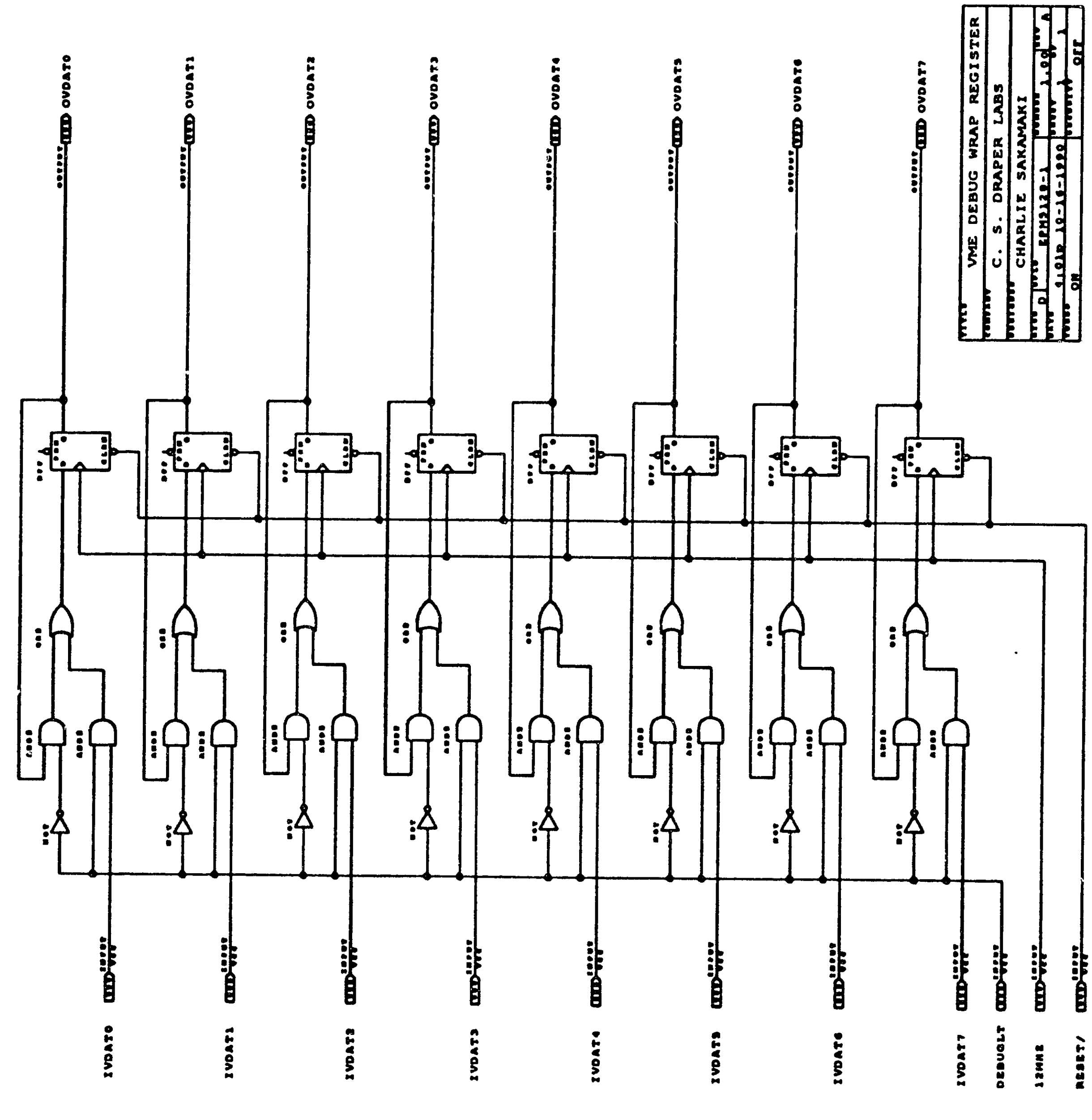
The primary conclusion of this thesis is that the Data Path Chip Set, comprising the FTC Chip and the Voter Chip, provides a compact and reliable mechanism for error detection/correction, synchronization, and communication functions needed for proper operation of the C3 FTPP. The use of FPGAs in this design proved to be cost-efficient, compact, and yet able to handle the clock speed of the design. Since the C3 FTPP is not to be manufactured in large quantities, it is sensible to use FPGAs for cost and flexibility reasons.

The C3 FTPP is expected to yield high performance and reliability. As the need for more powerful computers arises, several changes may be made to the C3 FTPP for enhanced performance and reliability. The Voter chip and the FTC chip may be combined into one chip when technology allows for that kind of density and pin count in an FPGA. One bottleneck in the C3 FTPP design is the inter-NE communication speed. This inter-communication speed of 12.5 million bytes per second is presently limited by the maximum clock speed of the AMD TAXI chips and IDT7202 FIFOs. To improve performance, these devices should be updated to those of a faster technology. This would lead to an increased clock speed throughout the NE, including both the Voter chip and FTC chips, thus requiring high speed FPGAs. Throughput may be increased by including more PEs, which would increase the complexity of the NE Global Controller and NE Scoreboard. Reliability may be increased by including more NEs and allowing more redundancy. For instance, there could be seven fully-interconnected NEs, which would Two-fault Byzantine Resilient. This would also call for three rounds of communication and increase the complexity of the design. Overall, the C3 FTPP is a highly flexible design, and the Data Path Chip Set may be easily modified to accommodate changes the C3 FTPP may go through in the years to come.

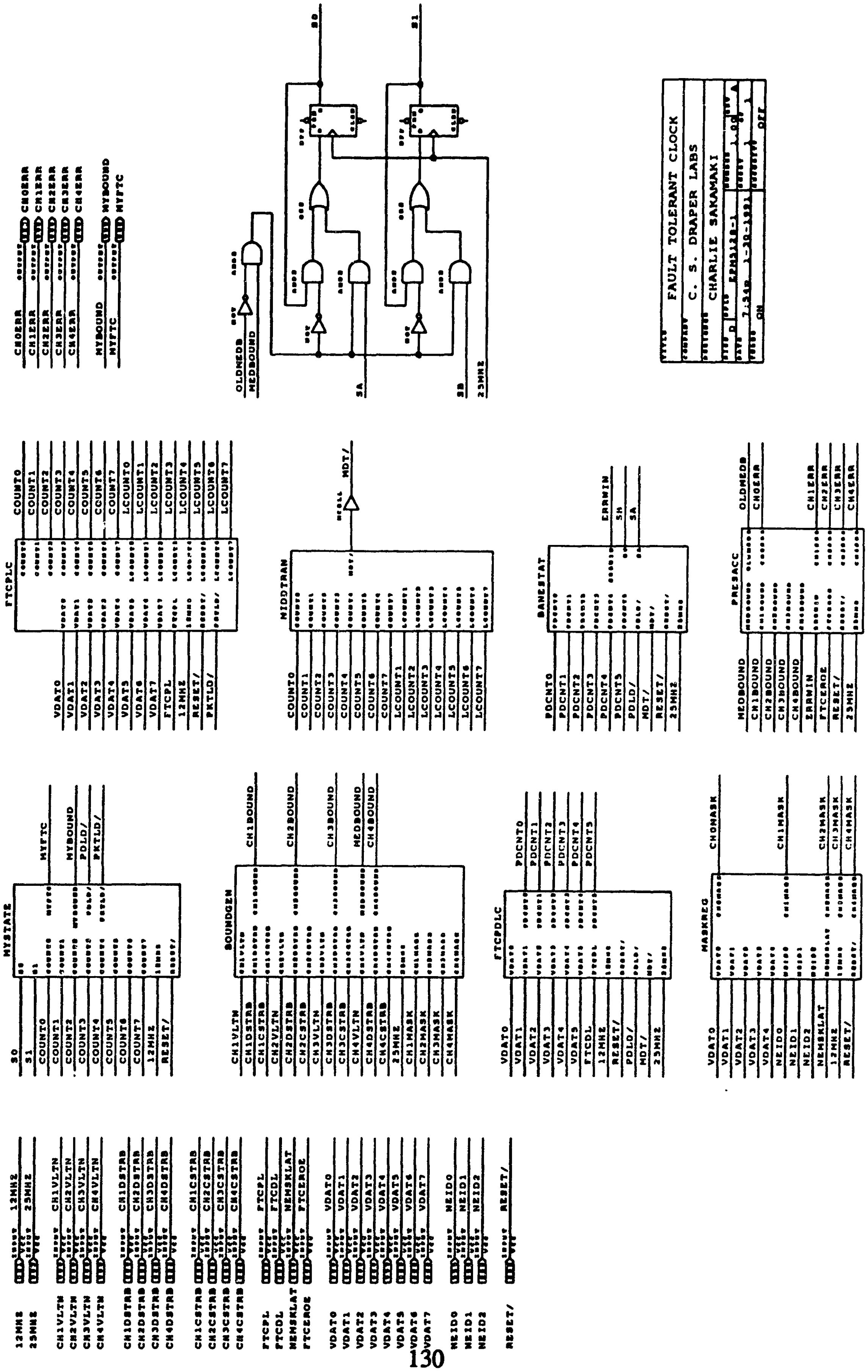
Appendix A - FTC Chip Schematics

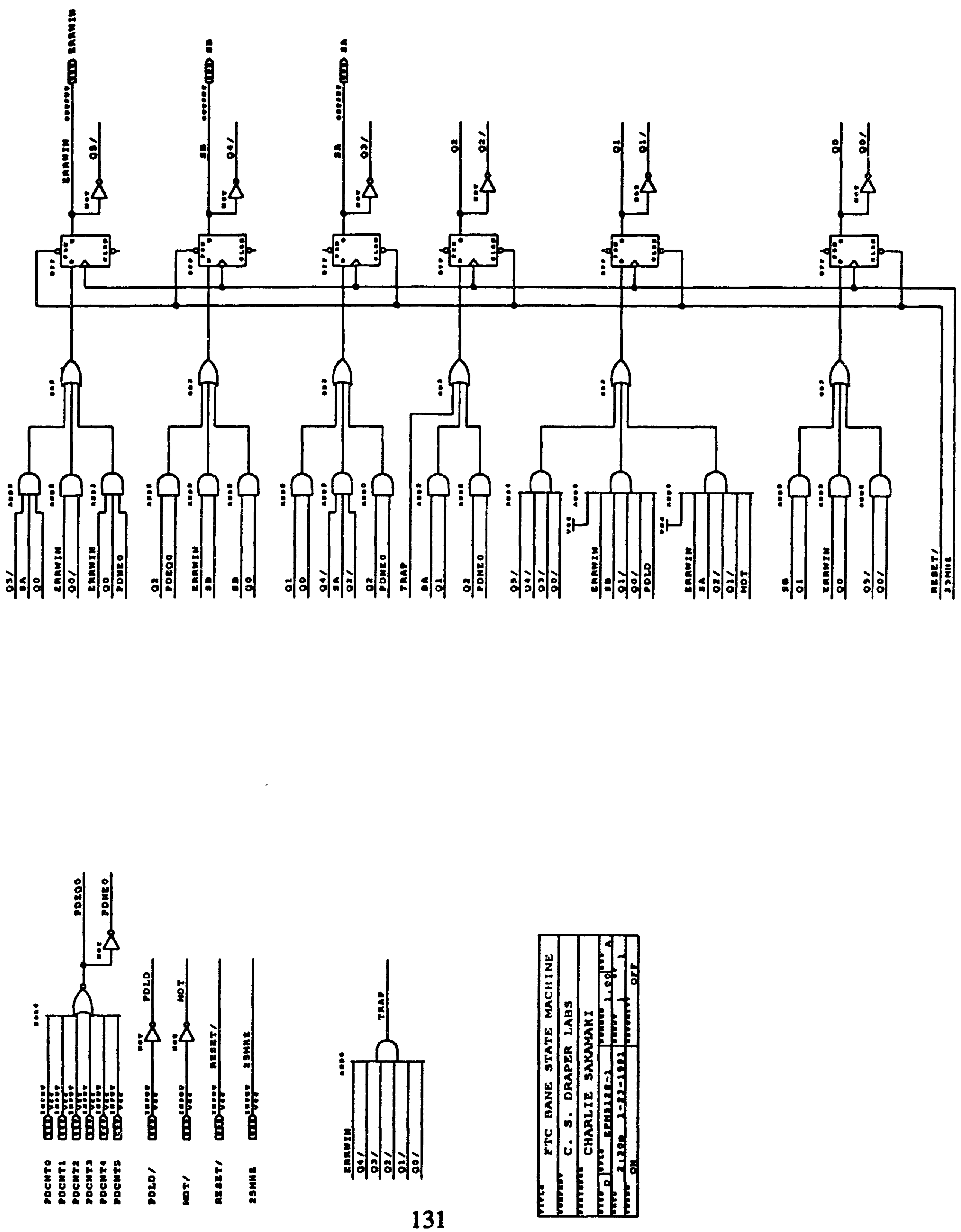


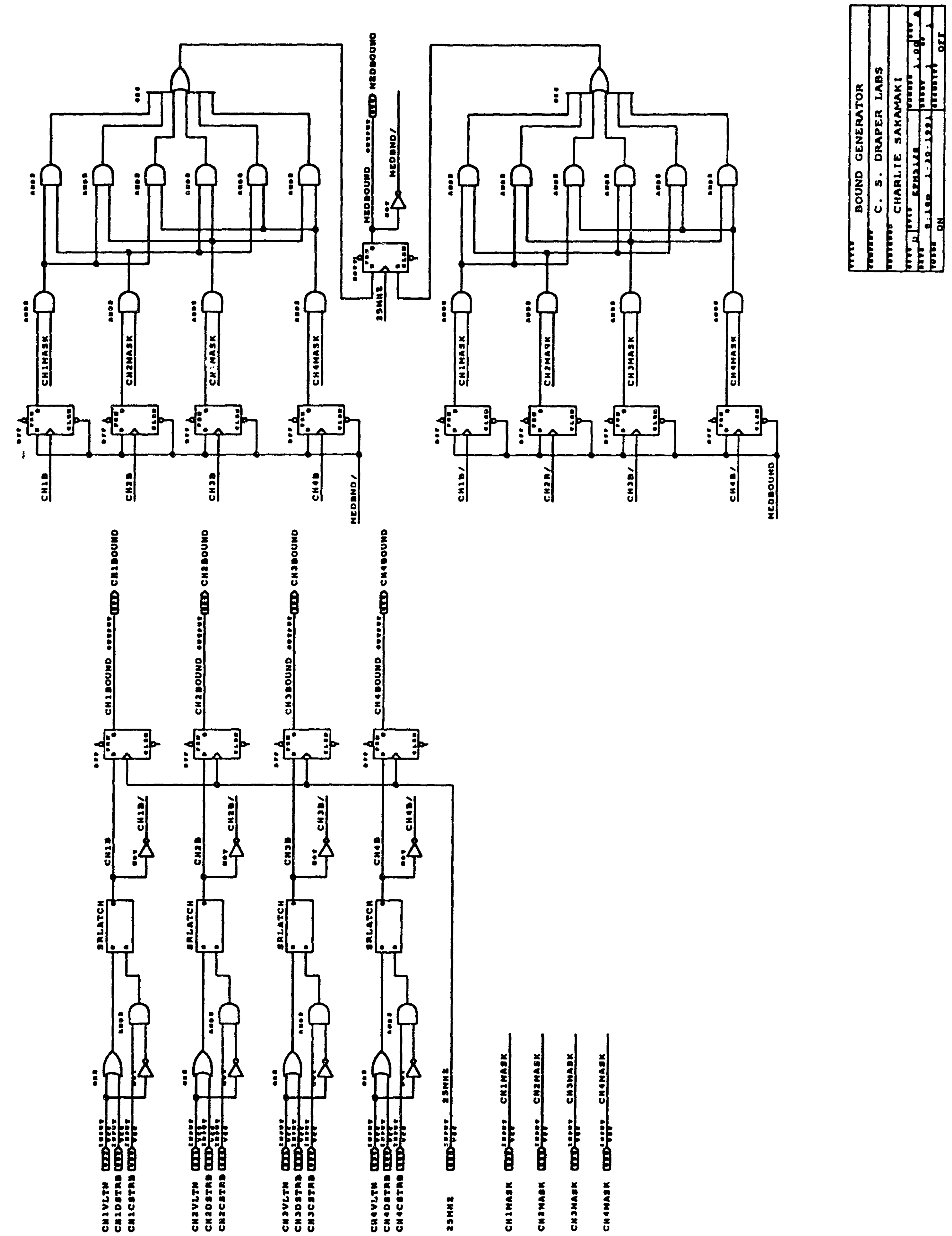

CH1SHIN/
CH2SHIN/
CH3SHIN/
CH4SHIN/
OUTPUT
OR3
AND3
AND4
AND2
CH1DSTRB
NEFUNC
CH1DBMD/
CH1CSTRB
CH1VLTN
CHOSH/
CH1DBMD
CH2DSTRB
CH2DBMD/
CH2CSTRB
CH2VLTN
CH2DBMD
CH3DSTRB
CH3DBMD/
CH3CSTRB
CH3VLTN
CH3DBMD
CH4DSTRB
CH4DBMD/
CH4CSTRB
CH4VLTN
CH4DBMD
INPUT
VCC
NOT
NEFUNC0
NEFUNC1
OR2
ASYNCHRONOUS CONTROL
C. S. DRAPER LABS
CHARLIE SAKAMAKI
EPM5128-1
1.00
A
7:51p 1-30-1991
1 OF 1
ON
OFF

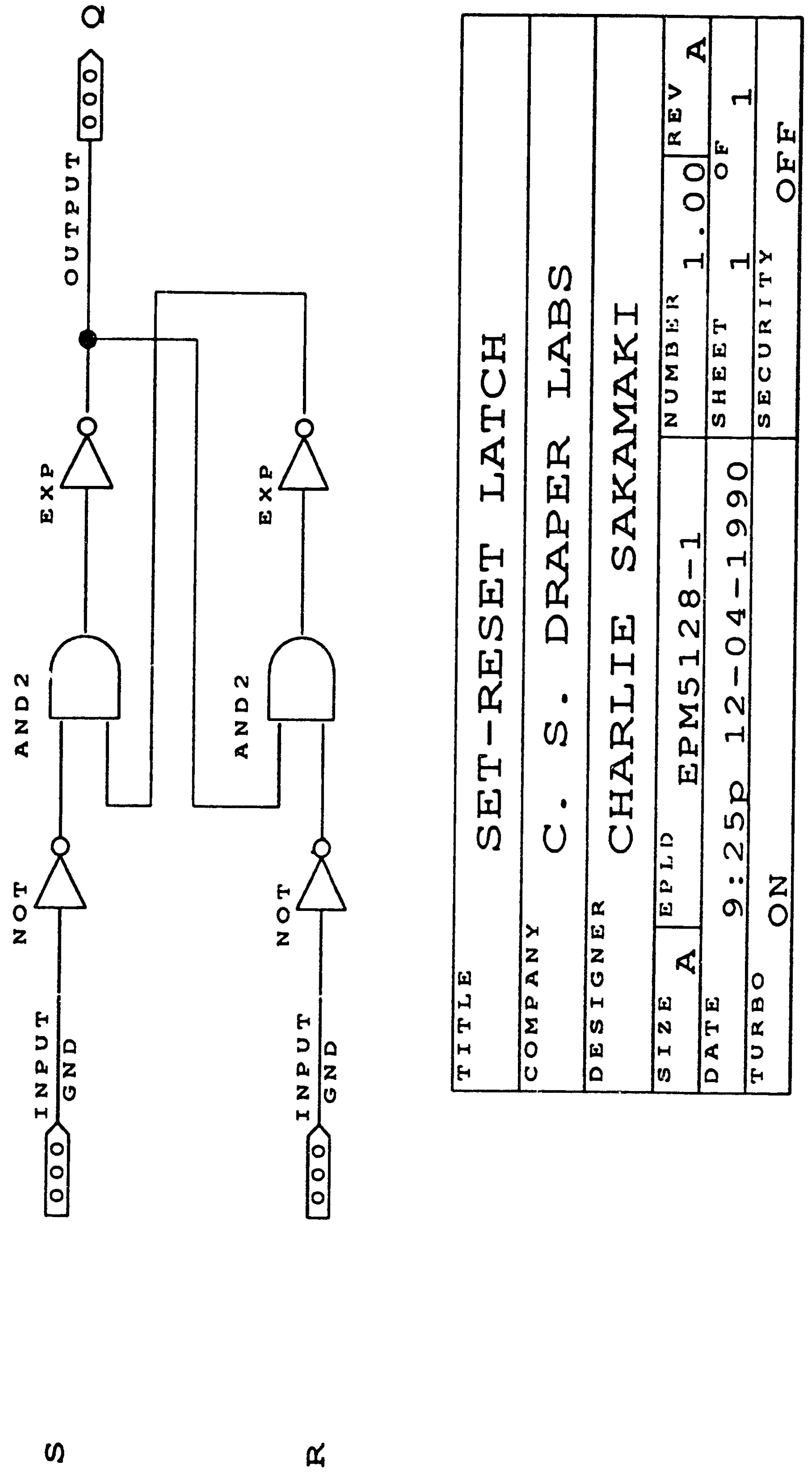

OVDAT0
OVDAT1
OVDAT2
OVDAT3
OVDAT4
OVDAT5
OVDAT6
OVDAT7
IVDAT0
IVDAT1
IVDAT2
IVDAT3
IVDAT4
IVDAT5
IVDAT6
IVDAT7
DEBUGLT
12MHZ
RESET/
VME DEBUG WRAP REGISTER
C. S. DRAPER LABS
CHARLIE SAKAMAKI

Appendix A - FTC Chip Schematics



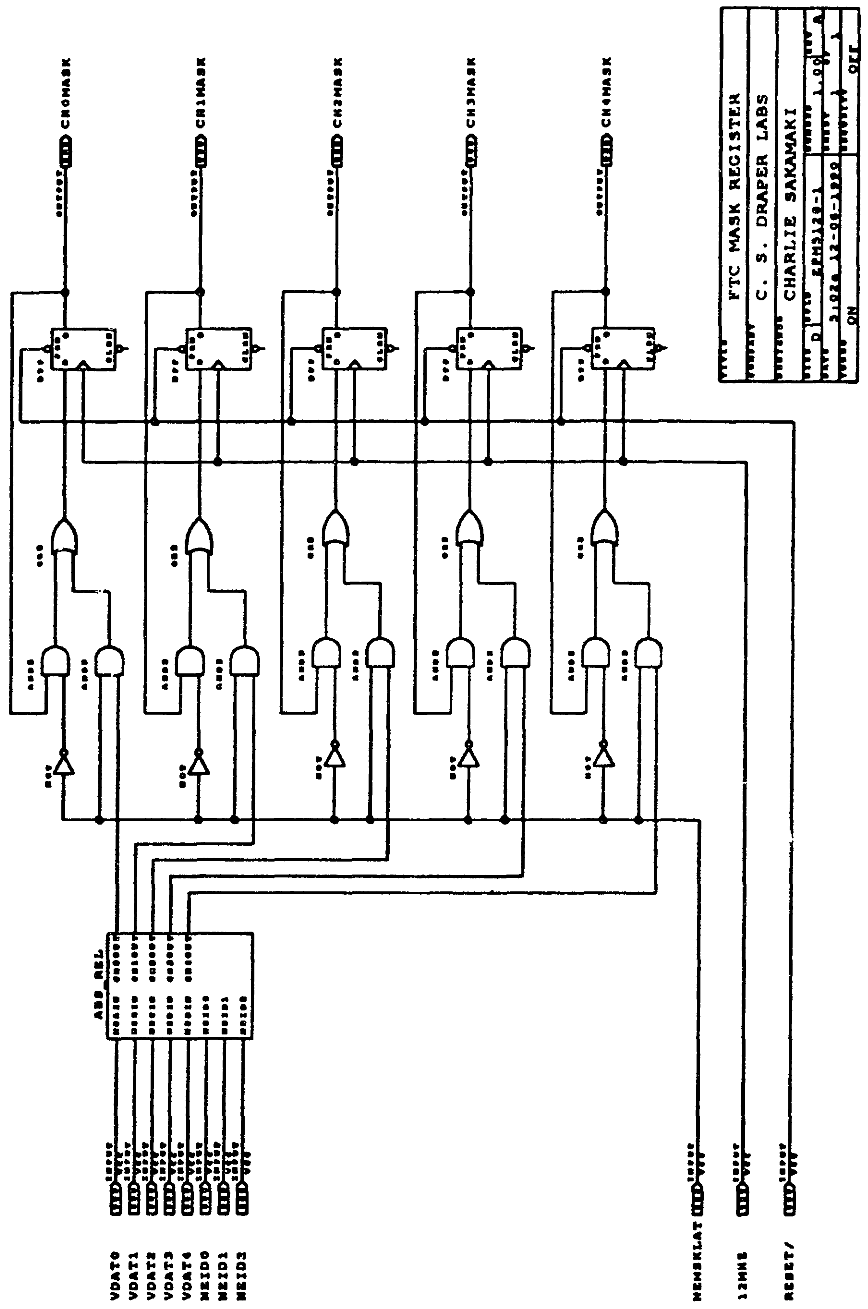

ERRWIN
SB
SA
Q2
Q1
Q0
Q5/
Q4/
Q3/
Q2/
Q1/
Q0/
TRAP
PDNEO
PDEQO
PDLD
HOT
RESET/
25MHZ
FTC BANE STATE MACHINE
C. S. DRAPER LABS
CHARLIE SAKAMAKI


BOUND GENERATOR
C. S. DRAPER LABS
CHARLIE SAKAMAKI
ON
OFF
MEDBOUND
MEDBND/
25MHZ
CH1MASK
CH2MASK
CH3MASK
CH4MASK
CH1B
CH2B
CH3B
CH4B
CH1B/
CH2B/
CH3B/
CH4B/
CH1BOUND
CH2BOUND
CH3BOUND
CH4BOUND
SRLATCH
CH1VLTN
CH1DSTRB
CH1CSTRB
CH2VLTN
CH2DSTRB
CH2CSTRB
CH3VLTN
CH3DSTRB
CH3CSTRB
CH4VLTN
CH4DSTRB
CH4CSTRB

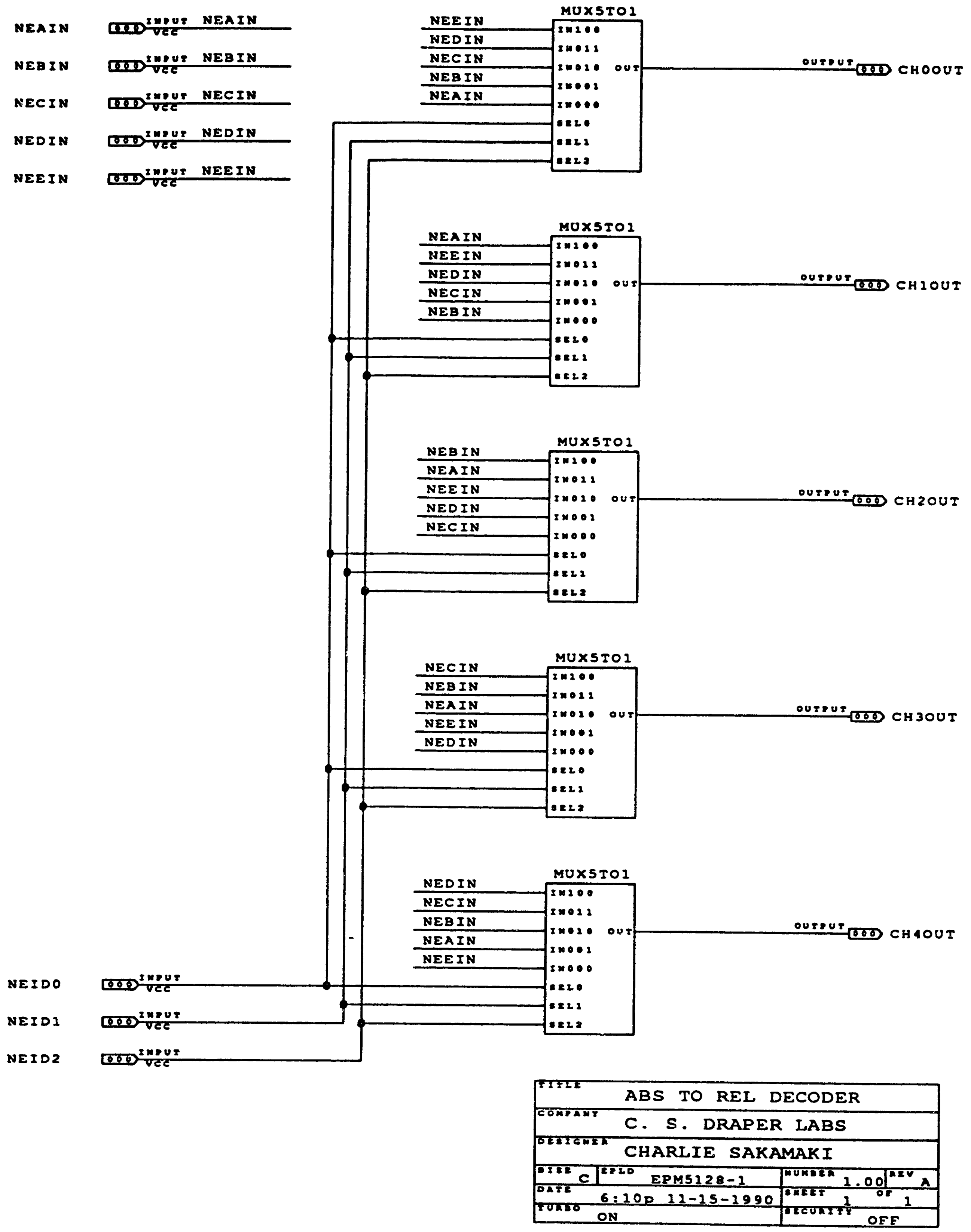


TITLE	SET-RESET LATCH
COMPANY	C. S. DRAPER LABS
DESIGNER	CHARLIE SAKAMAKI
SIZE A	EPLD EPM5128-1 / NUMBER 1.00 / REV A
DATE	9:25p 12-04-1990 / SHEET 1 OF 1
TURBO ON	SECURITY OFF

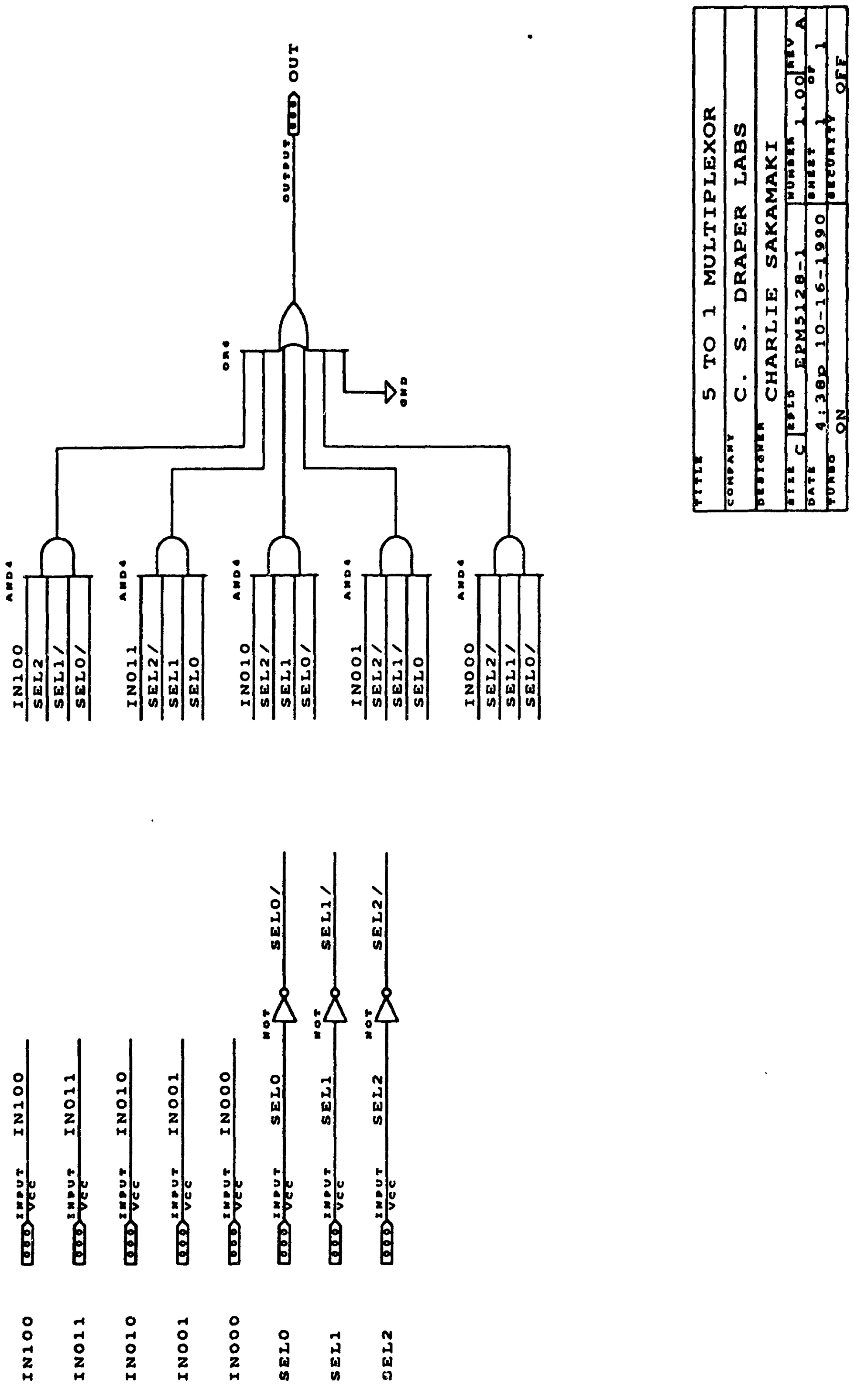
Appendix A - FTC Chip Schematics



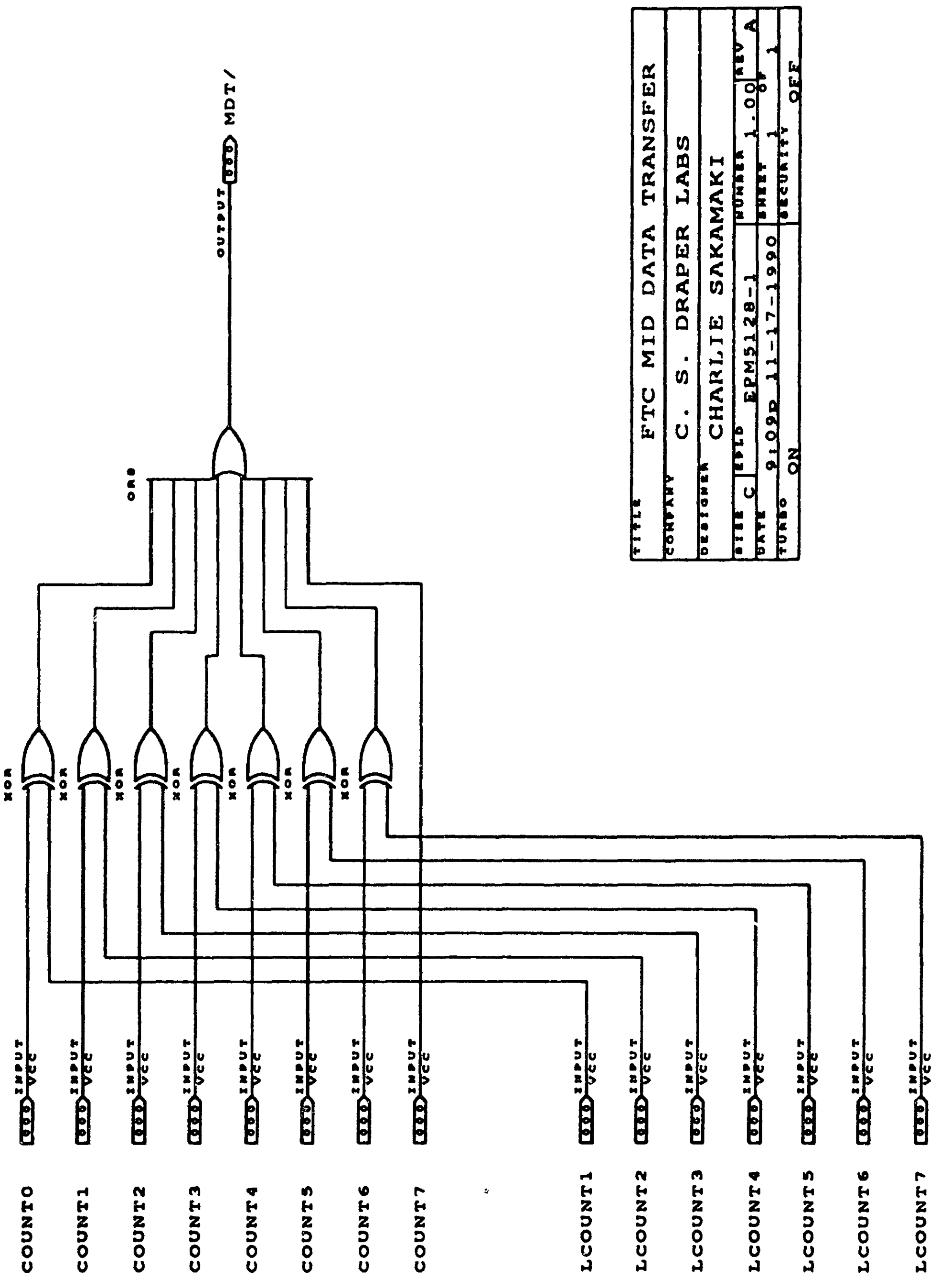
Appendix A - FTC Chip Schematics



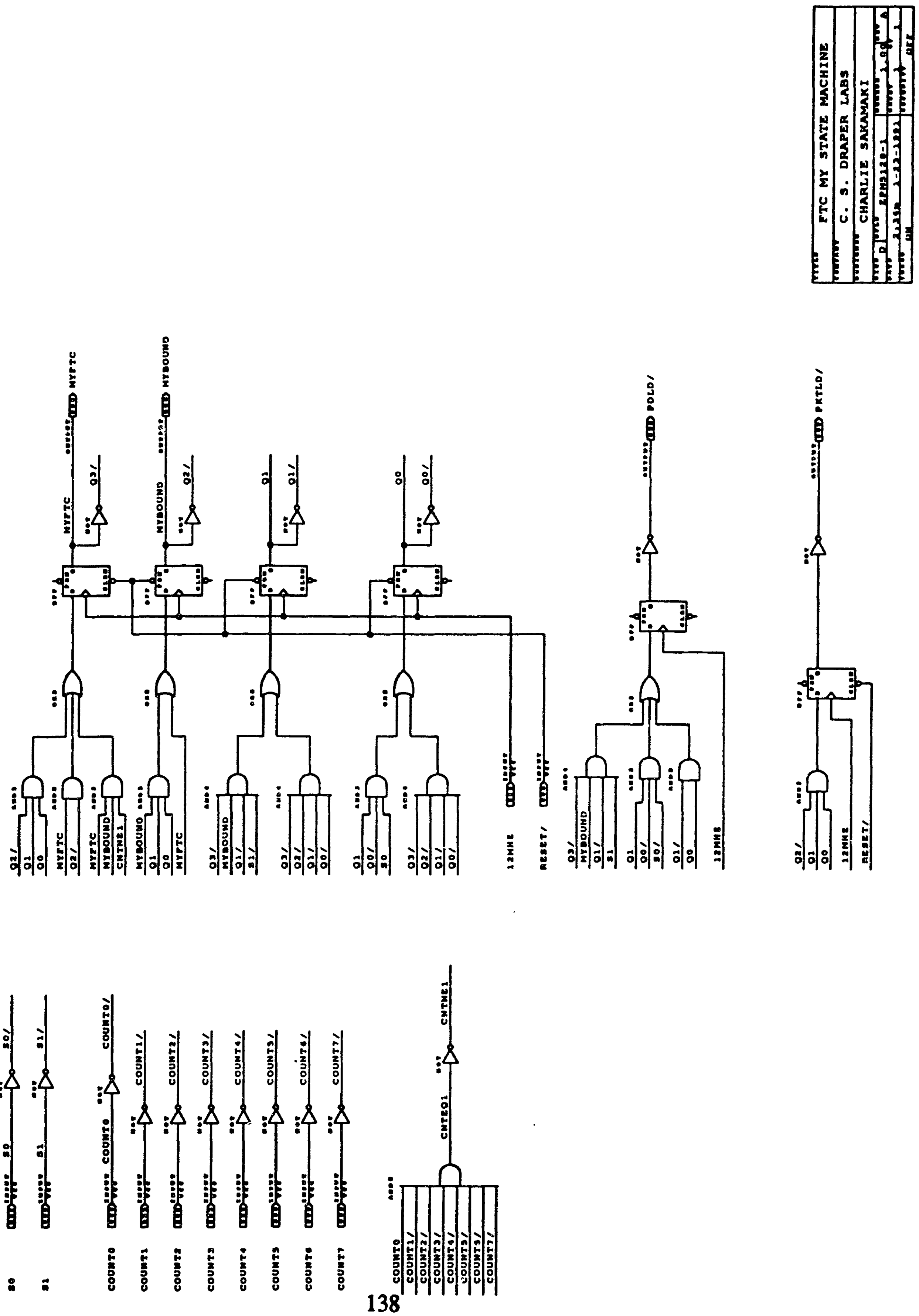
Appendix A - FTC Chip Schematics



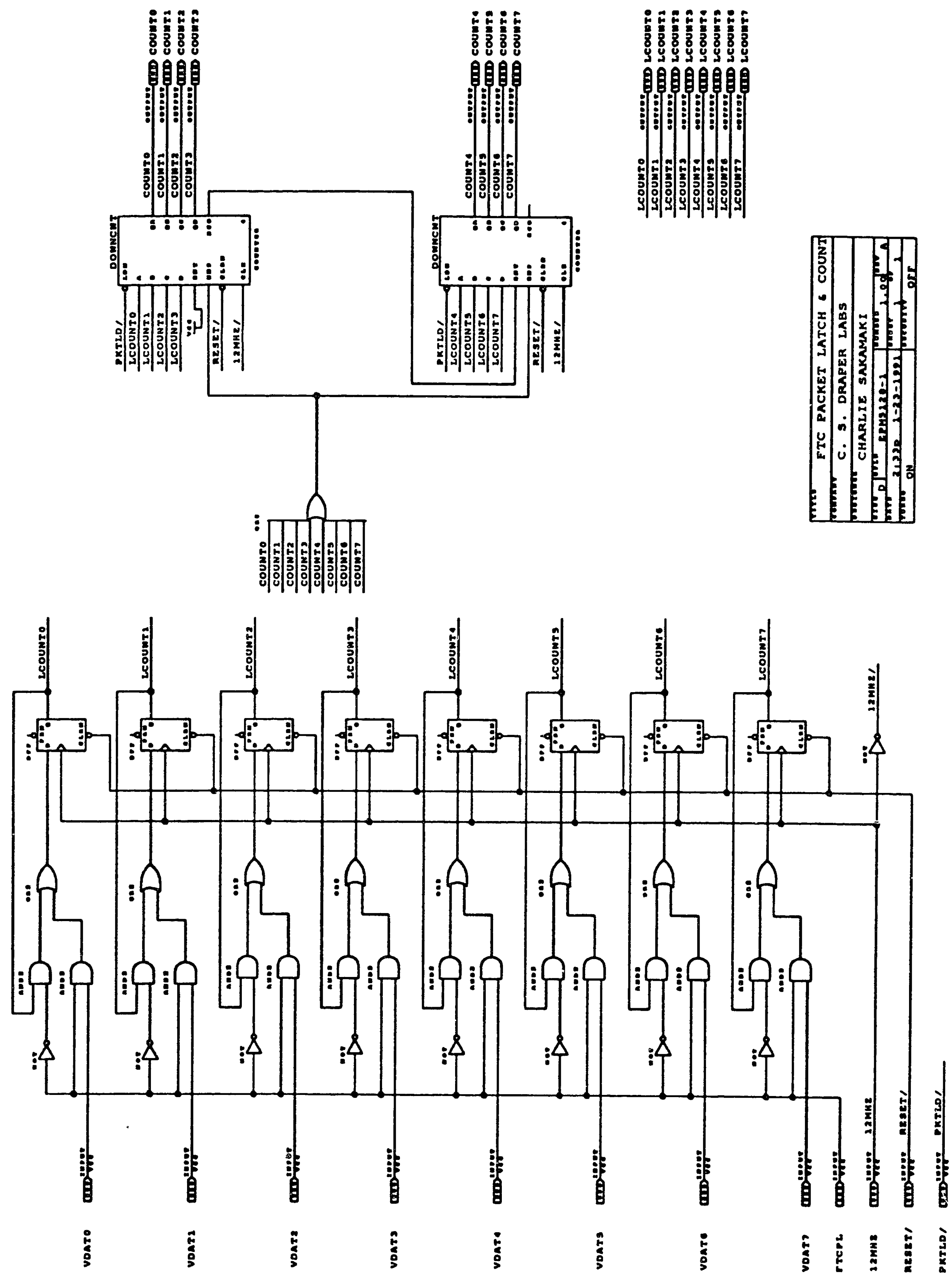
Appendix A - FTC Chip Schematics



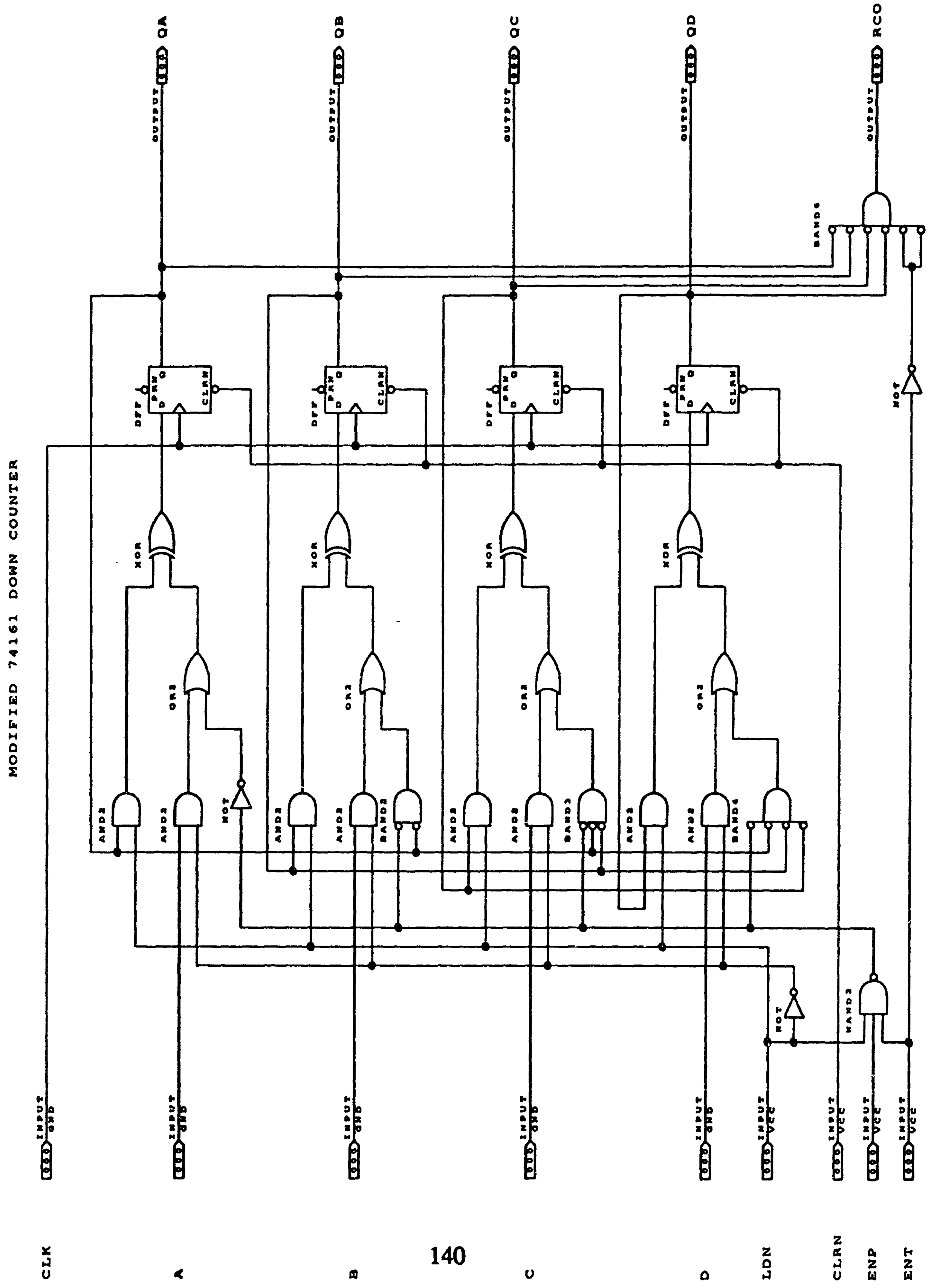
Appendix A - FTC Chip Schematics



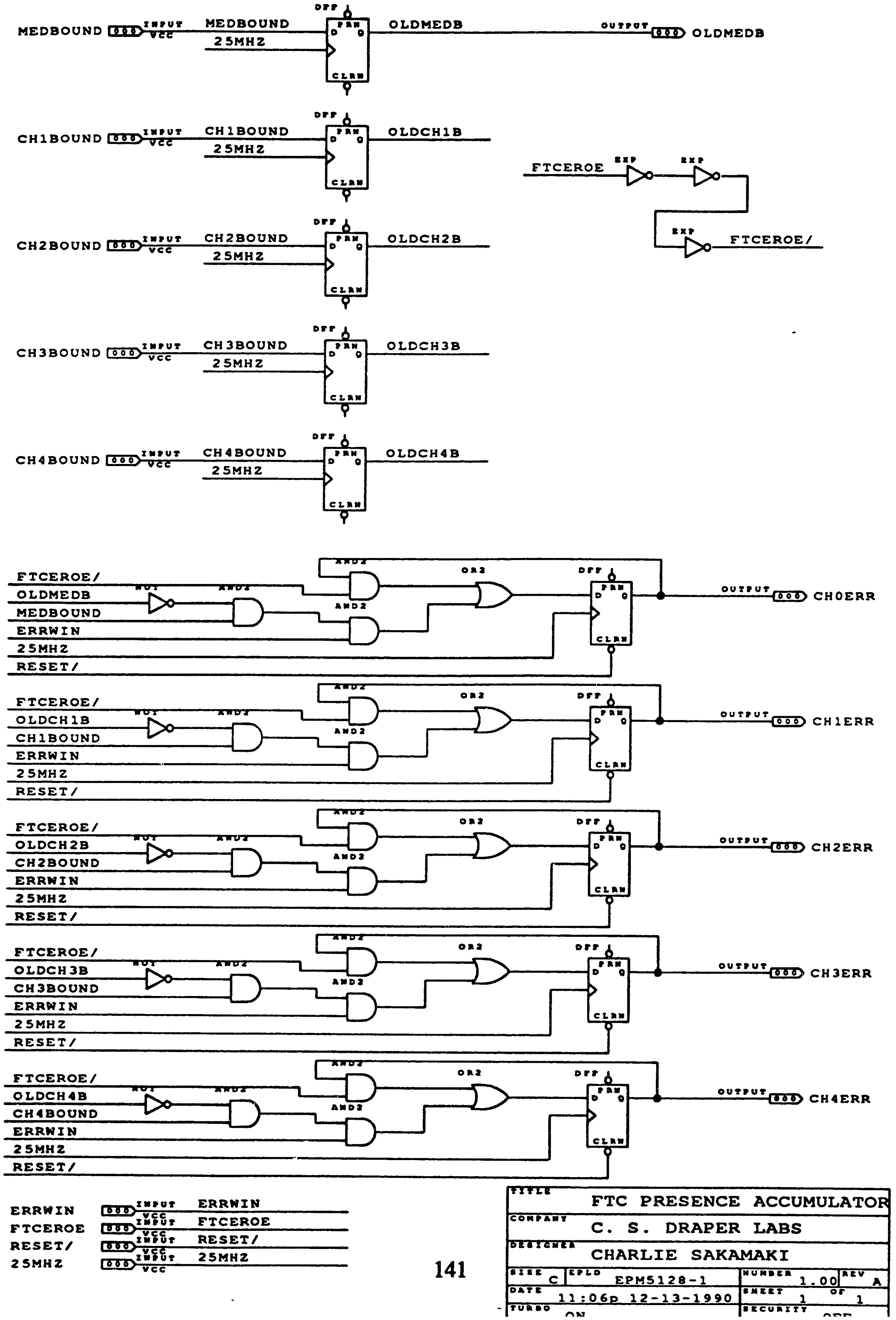
Appendix A - FTC Chip Schematics



Appendix A - FTC Chip Schematics

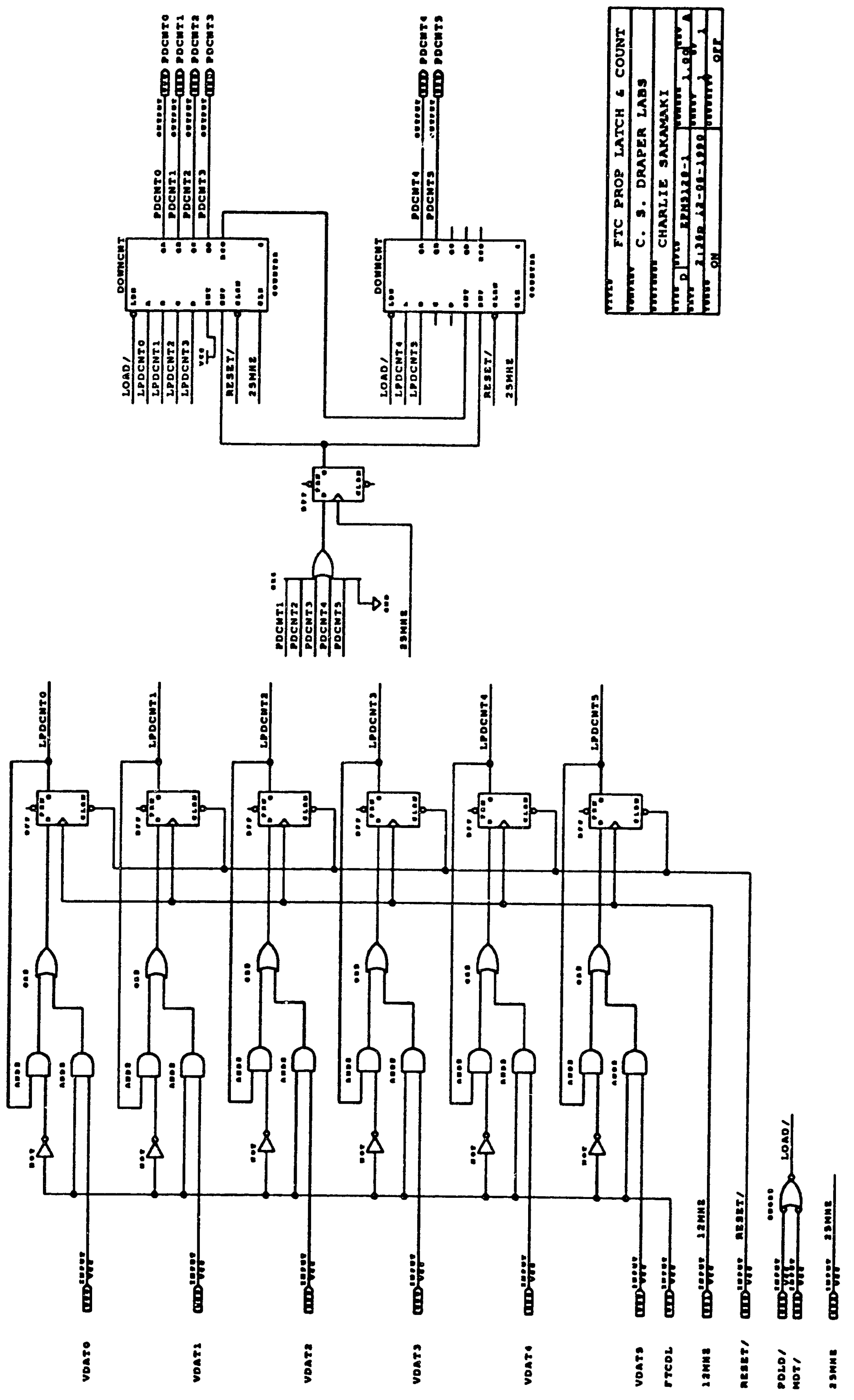


Appendix A - FTC Chip Schematics

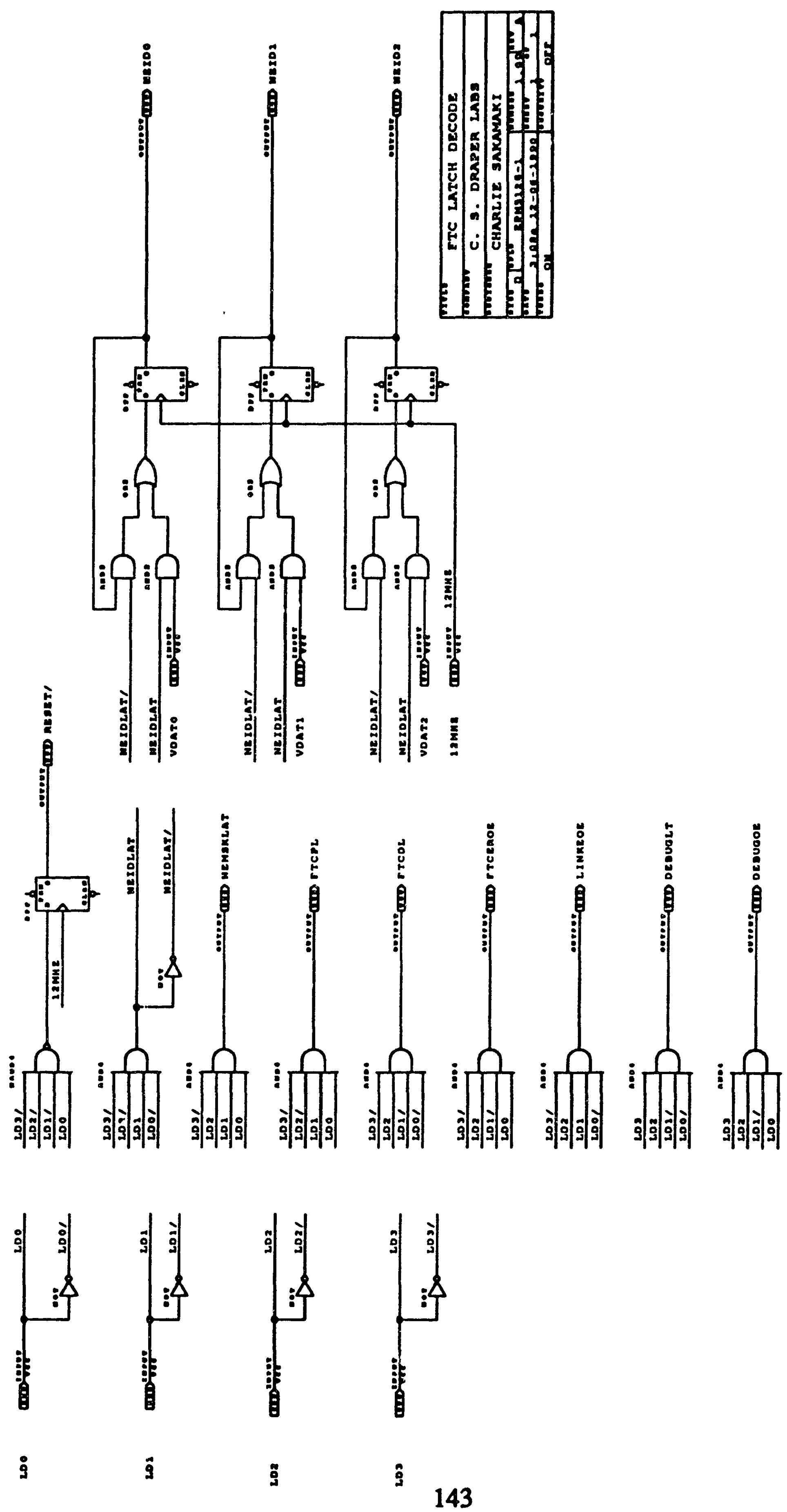


TITLE	FTC PRESENCE ACCUMULATOR
COMPANY	C. S. DRAPER LABS
DESIGNER	CHARLIE SAKAMAKI
SIZE C	EPLD EPM5128-1
NUMBER	1.00
REV	A
DATE	11:06p 12-13-1990
SHEET	1 OF 1
TURBO	ON
SECURITY	OFF

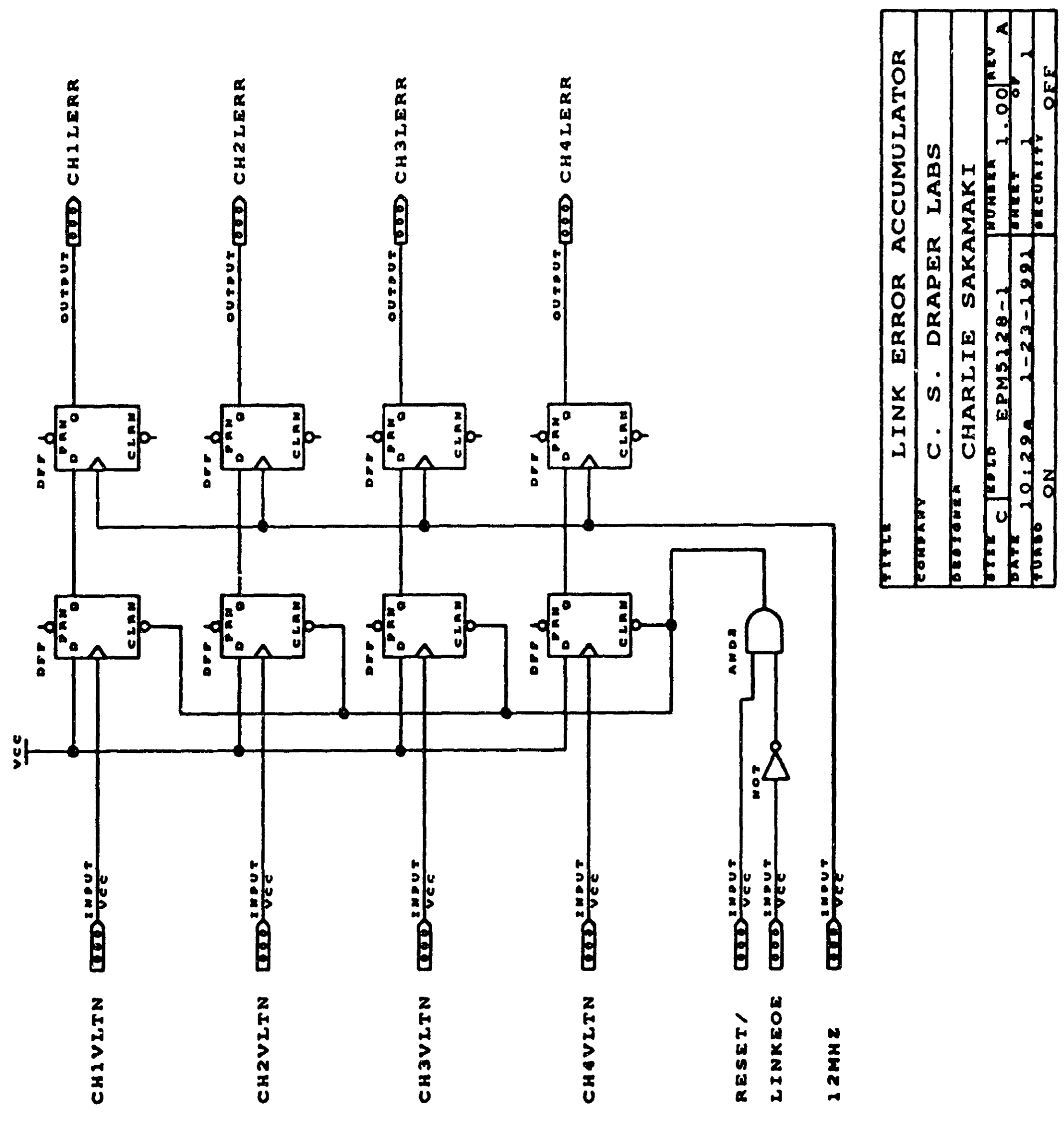
Appendix A - FTC Chip Schematics



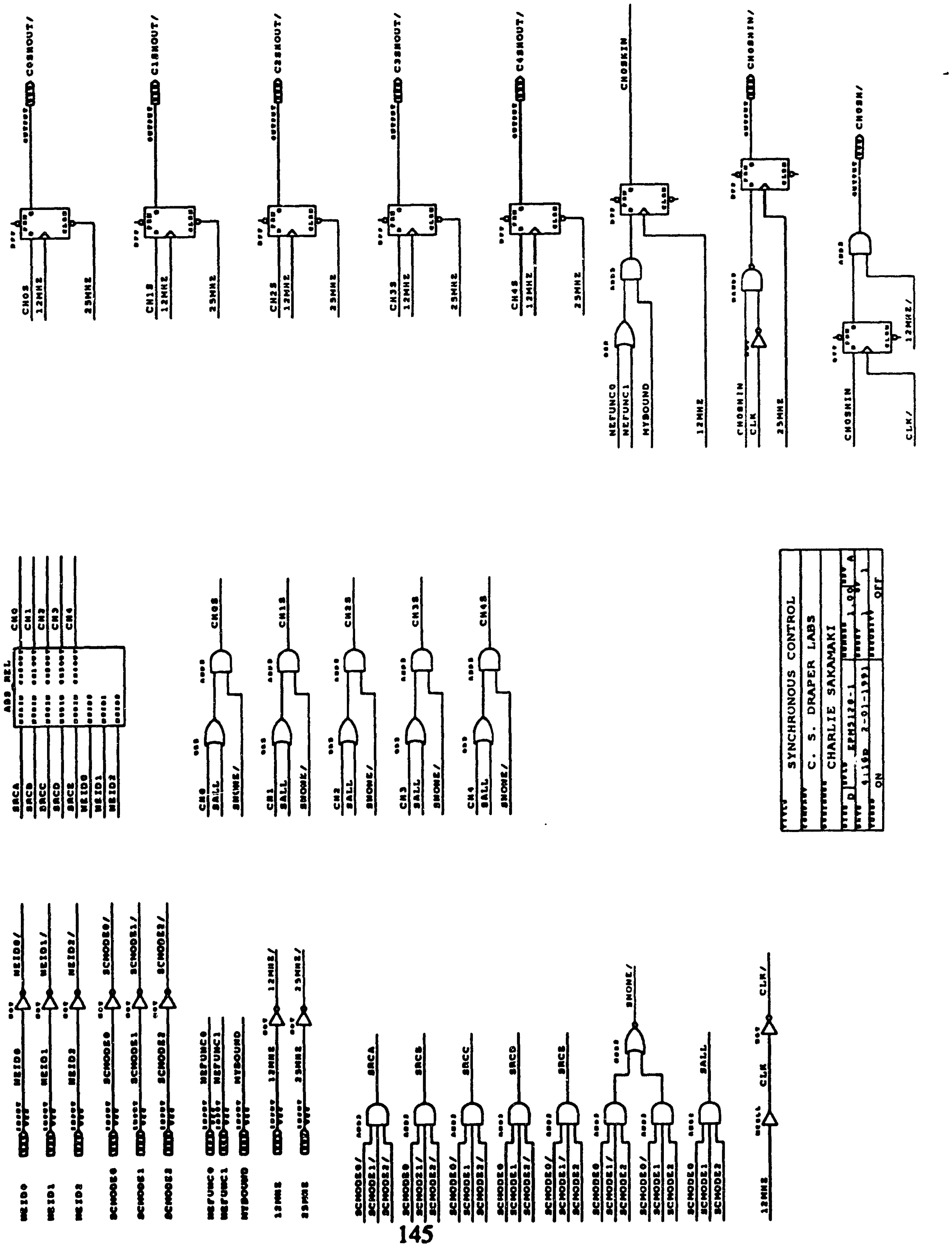
Appendix A - FTC Chip Schematics



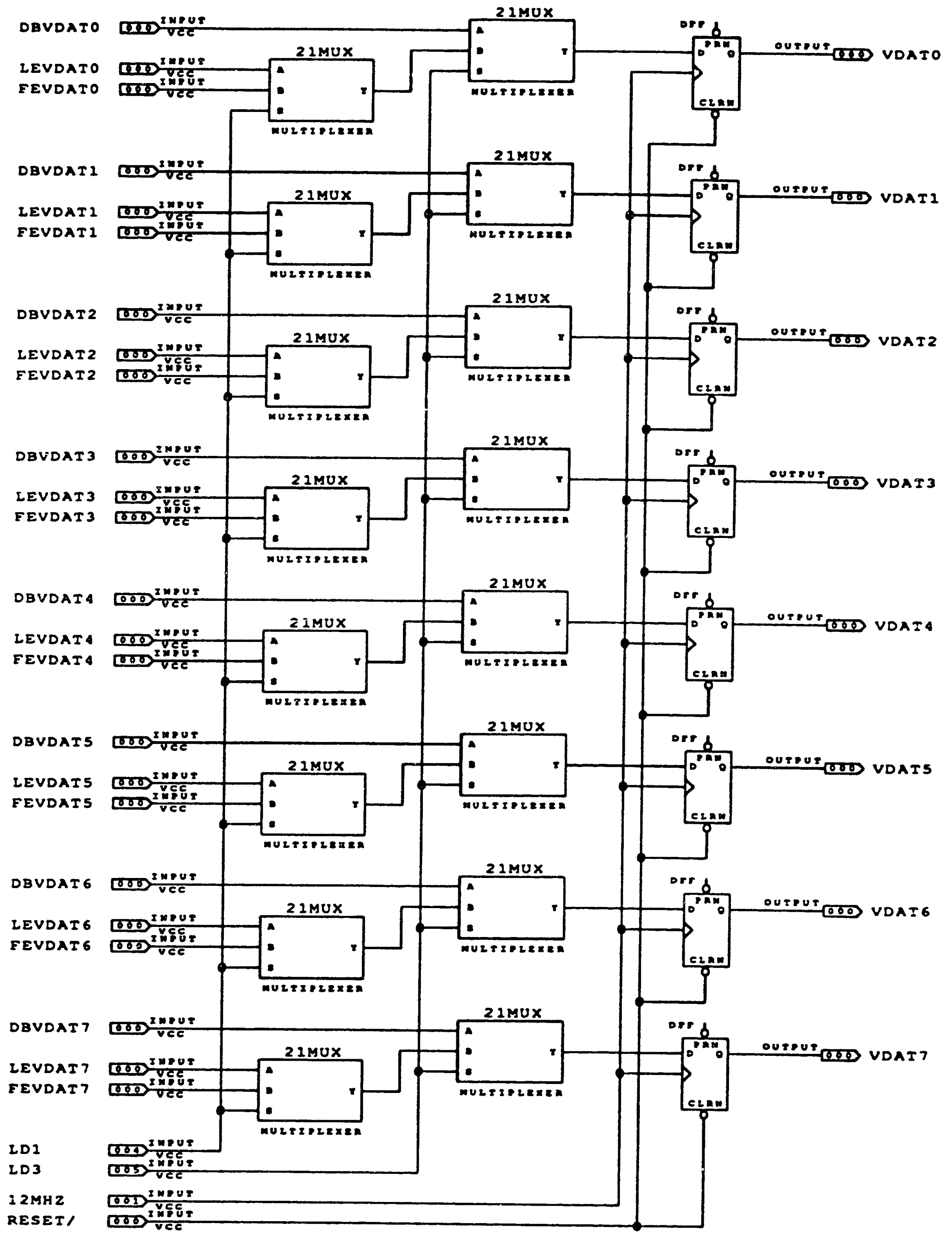
Appendix A - FTC Chip Schematics


CH1VLTN
CH2VLTN
CH3VLTN
CH4VLTN
RESET/
LINKEOE
12MHZ
INPUT
VCC
NOT
AND2
DFF
PRN
CLRN
OUTPUT
CH1LERR
CH2LERR
CH3LERR
CH4LERR
TITLE LINK ERROR ACCUMULATOR
COMPANY C. S. DRAPER LABS
DESIGNER CHARLIE SAKAMAKI
SIZE C EPLD EPM5128-1 NUMBER 1.00 REV A
DATE 10:29a 1-23-1991 SHEET 1 OF 1
TURBO ON SECURITY OFF

Appendix A - FTC Chip Schematics

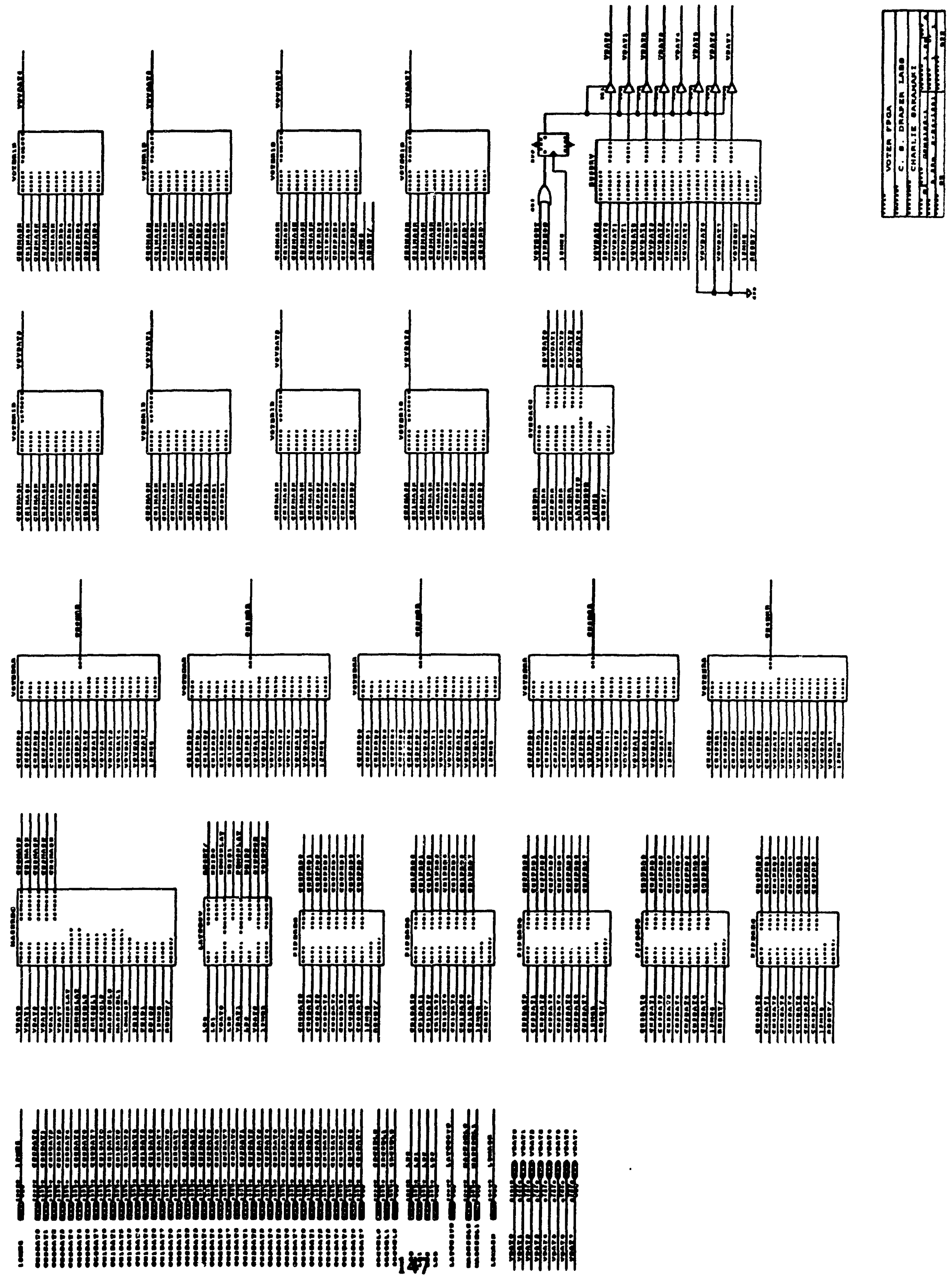


Appendix A - FTC Chip Schematics

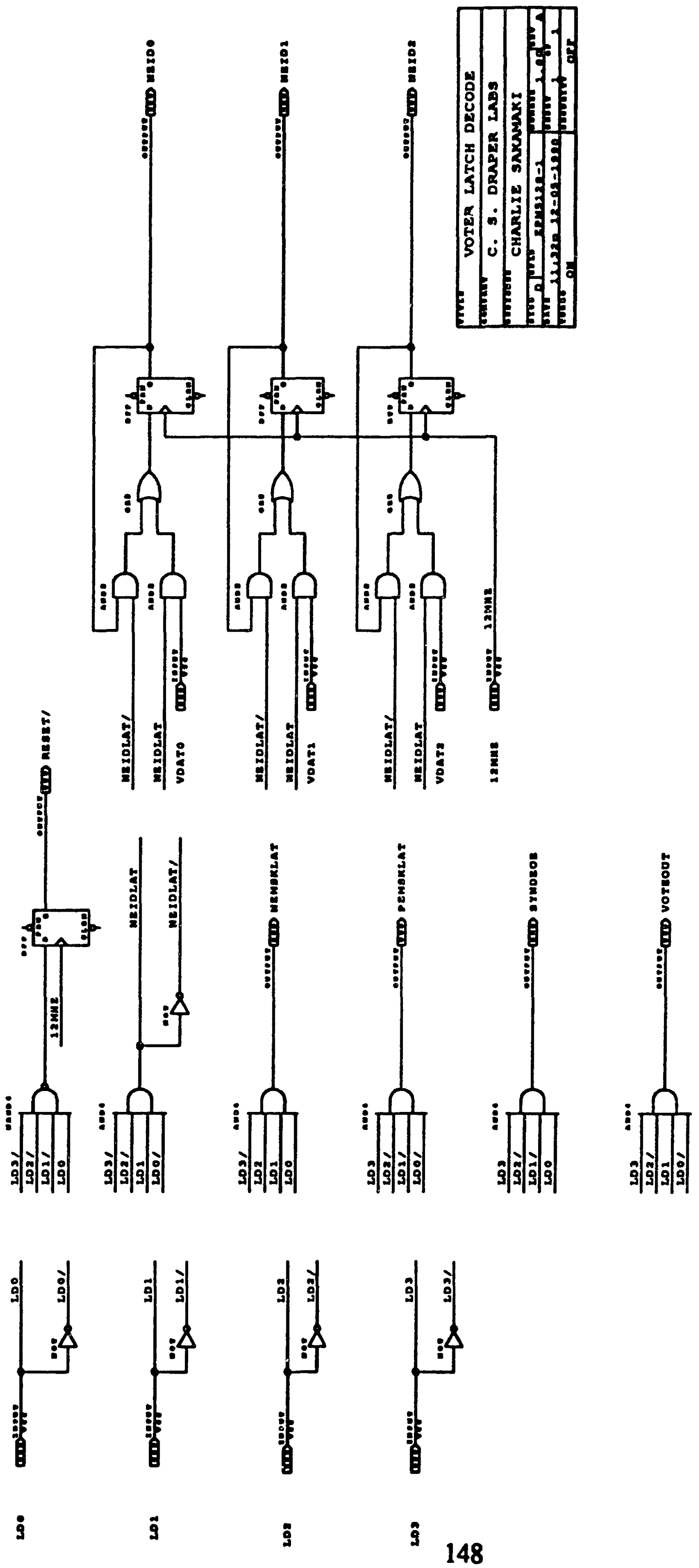


TITLE	FTC CHIP BUS DRIVER		
COMPANY	C. S. DRAPER LABS		
DESIGNER	CHARLIE SAKAMAKI		
SIZE C	EPLD EPM5128-1	NUMBER 1.00	REV A
DATE	10:30p 12-13-1990	SHEET 1 OF 1	
TURBO	ON	SECURITY OFF	

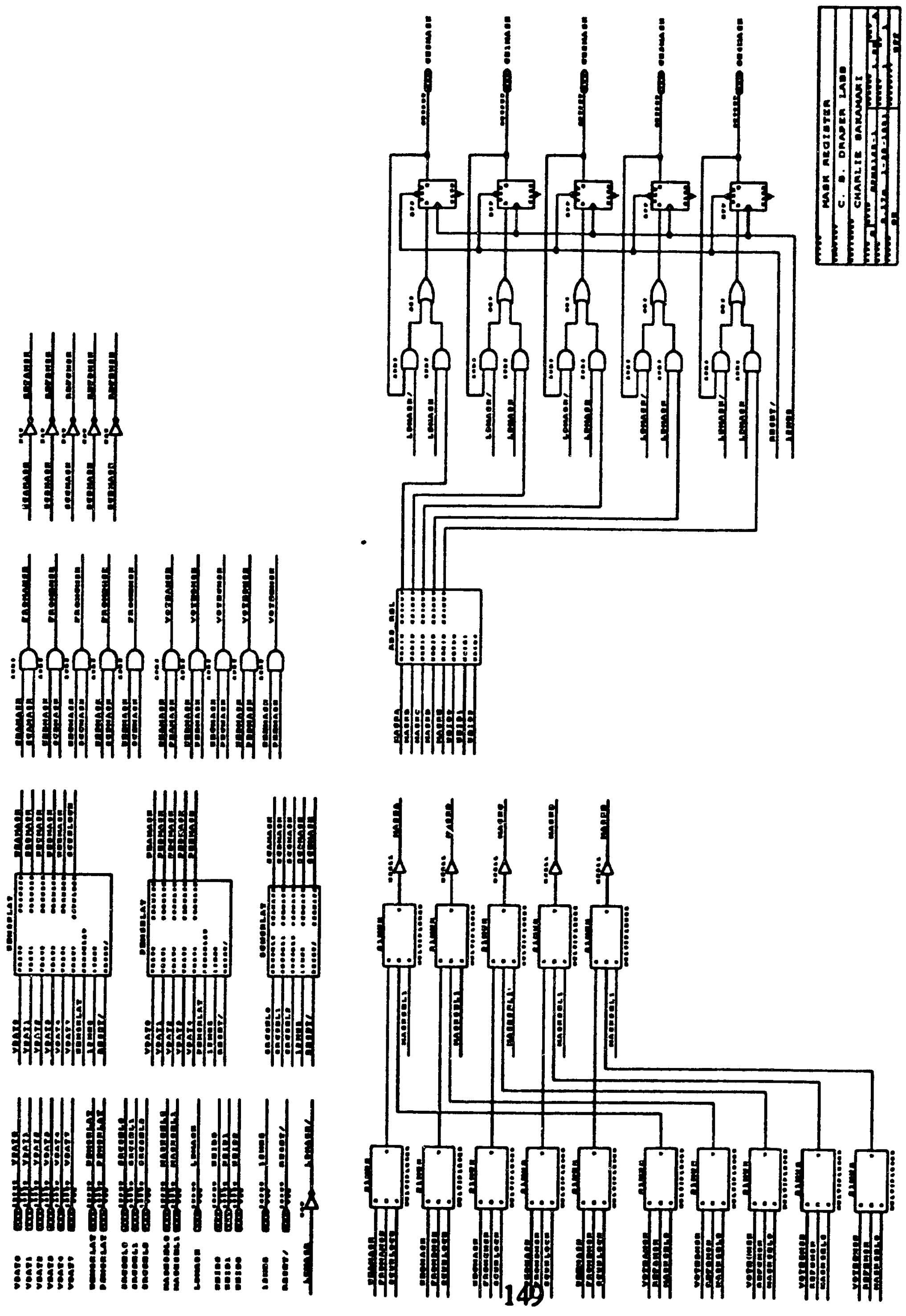
Appendix B - Voter Chip Schematics



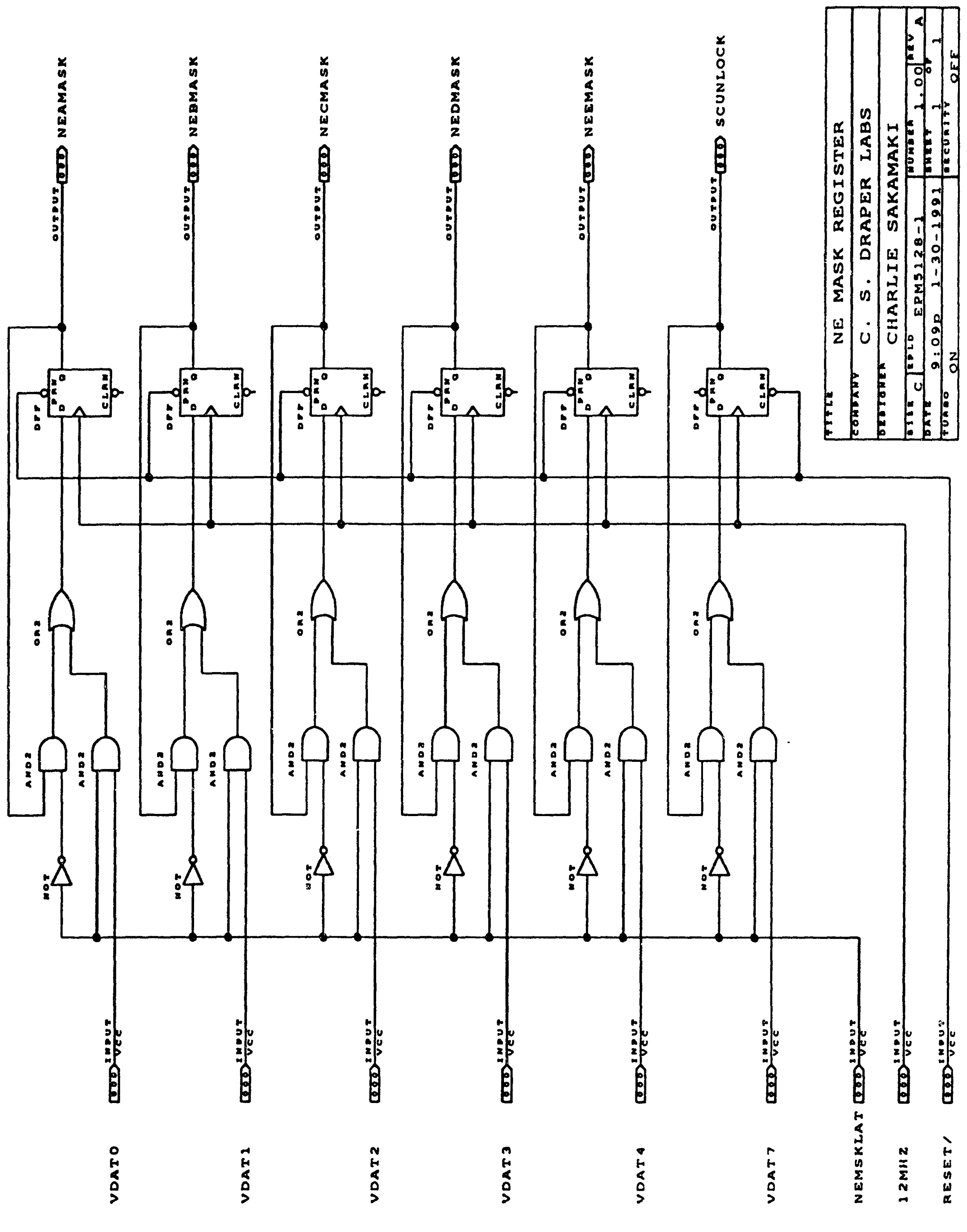
Appendix B - Voter Chip Schematics



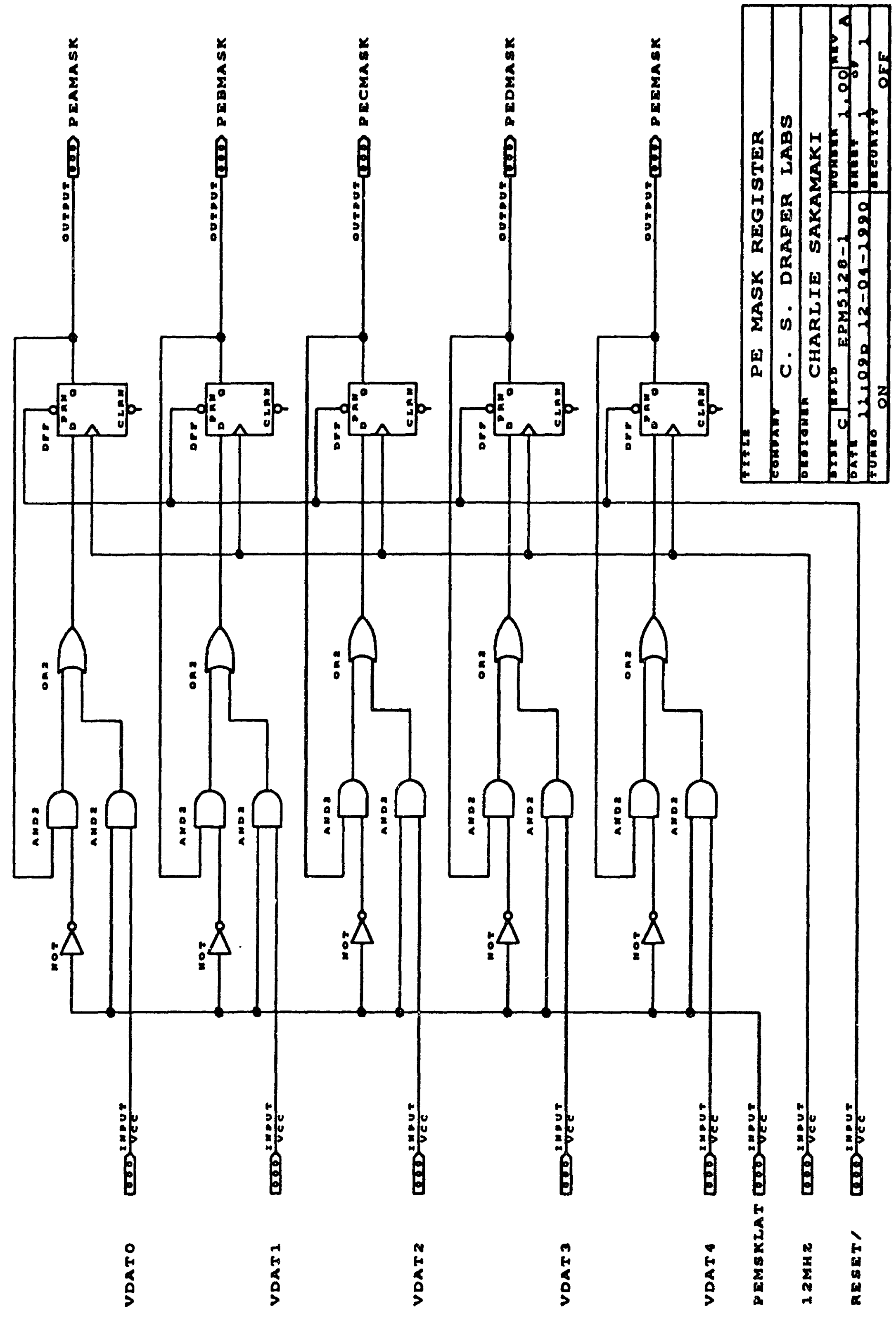
Appendix B - Voter Chip Schematics



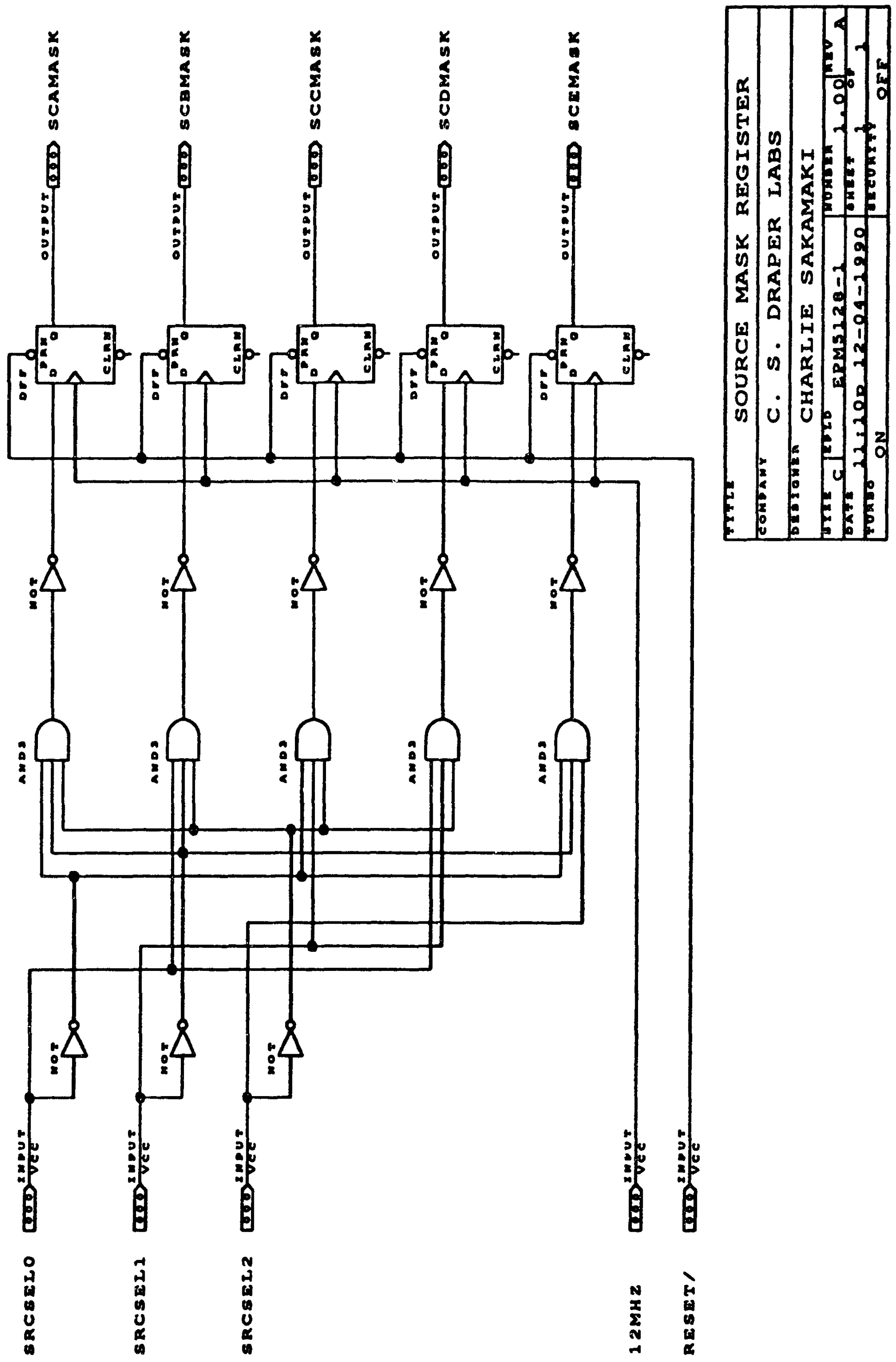
Appendix B - Voter Chip Schematics



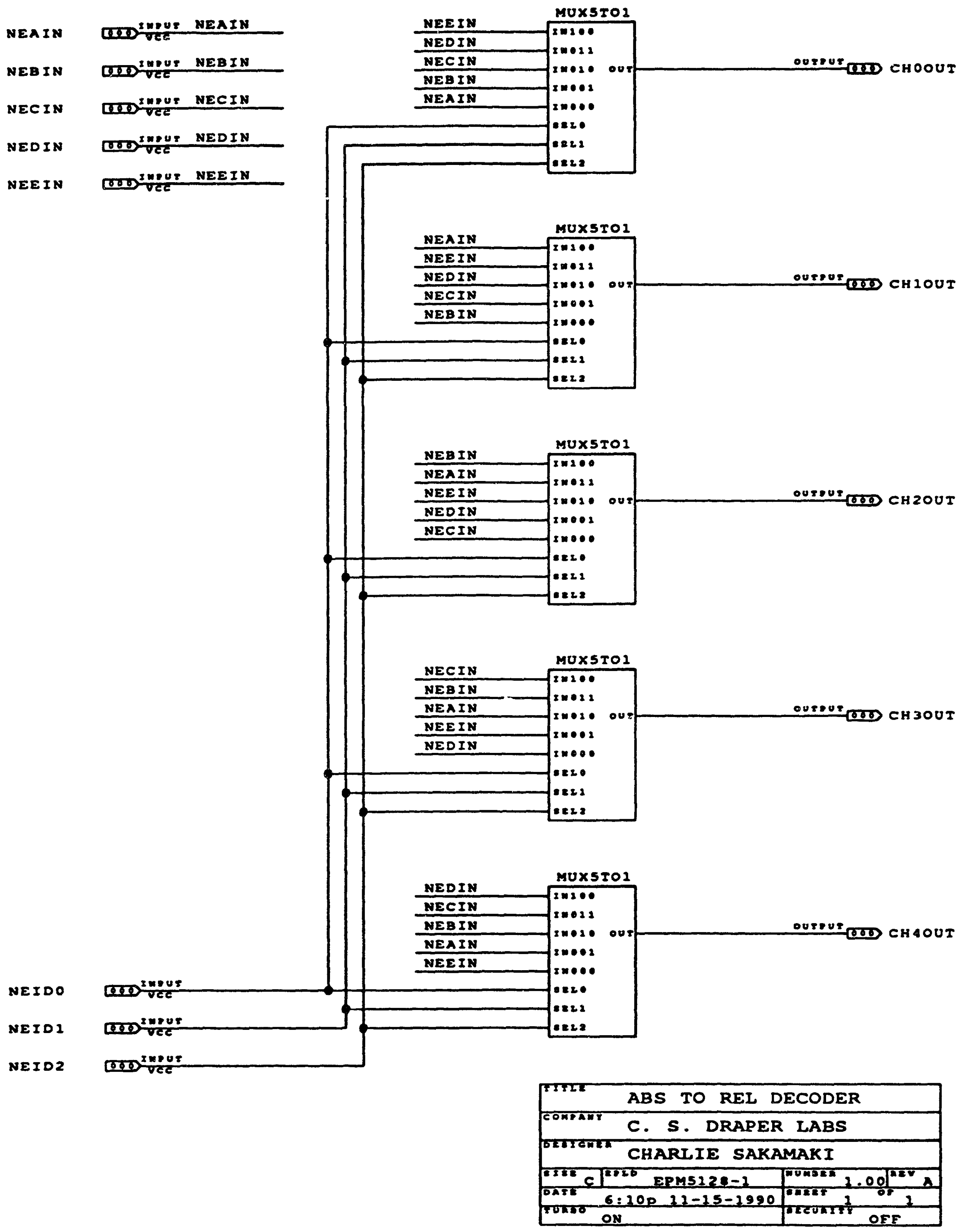
Appendix B - Voter Chip Schematics

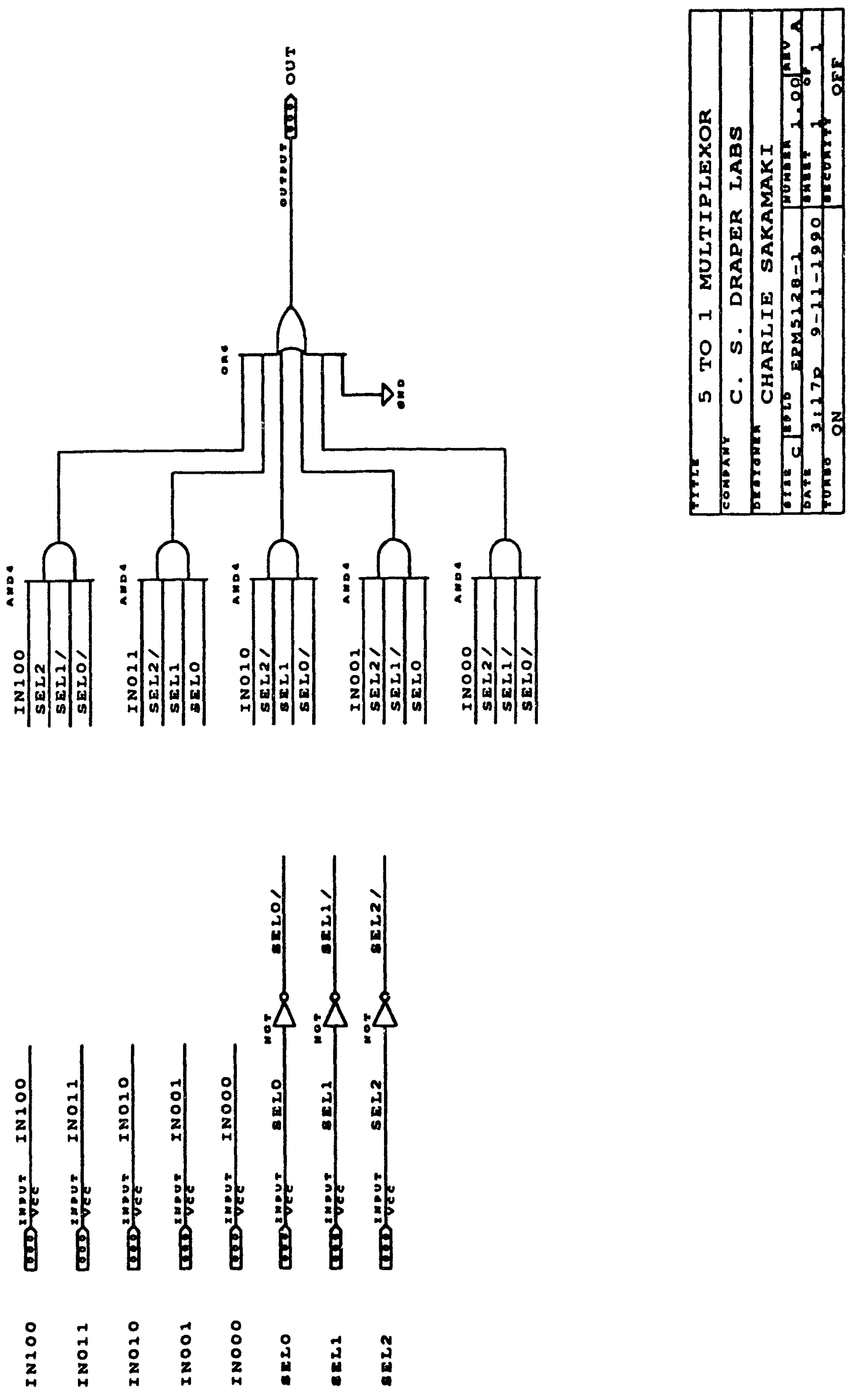


Appendix B - Voter Chip Schem[illegible]

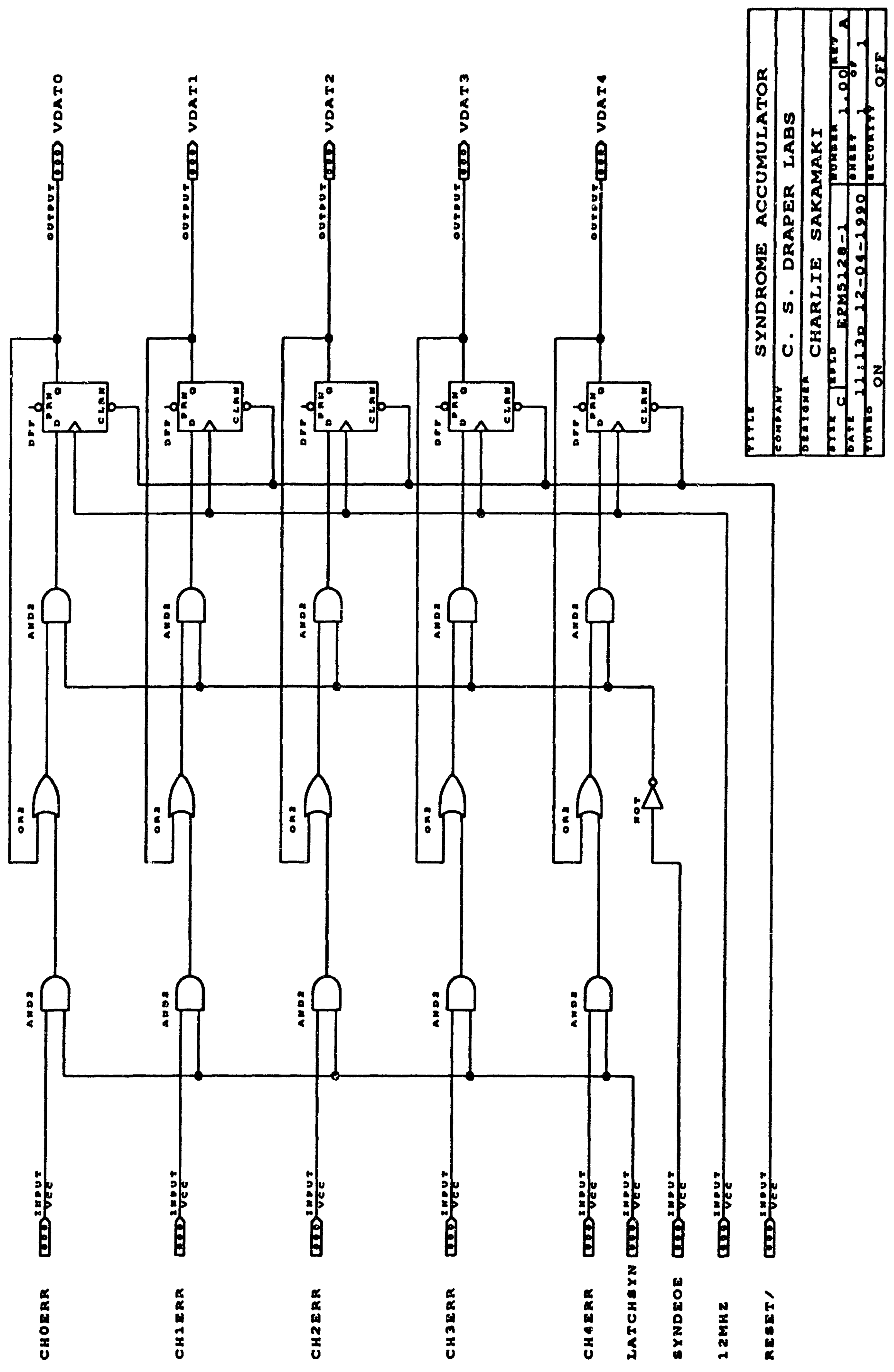


Appendix B - Voter Chip Schematics

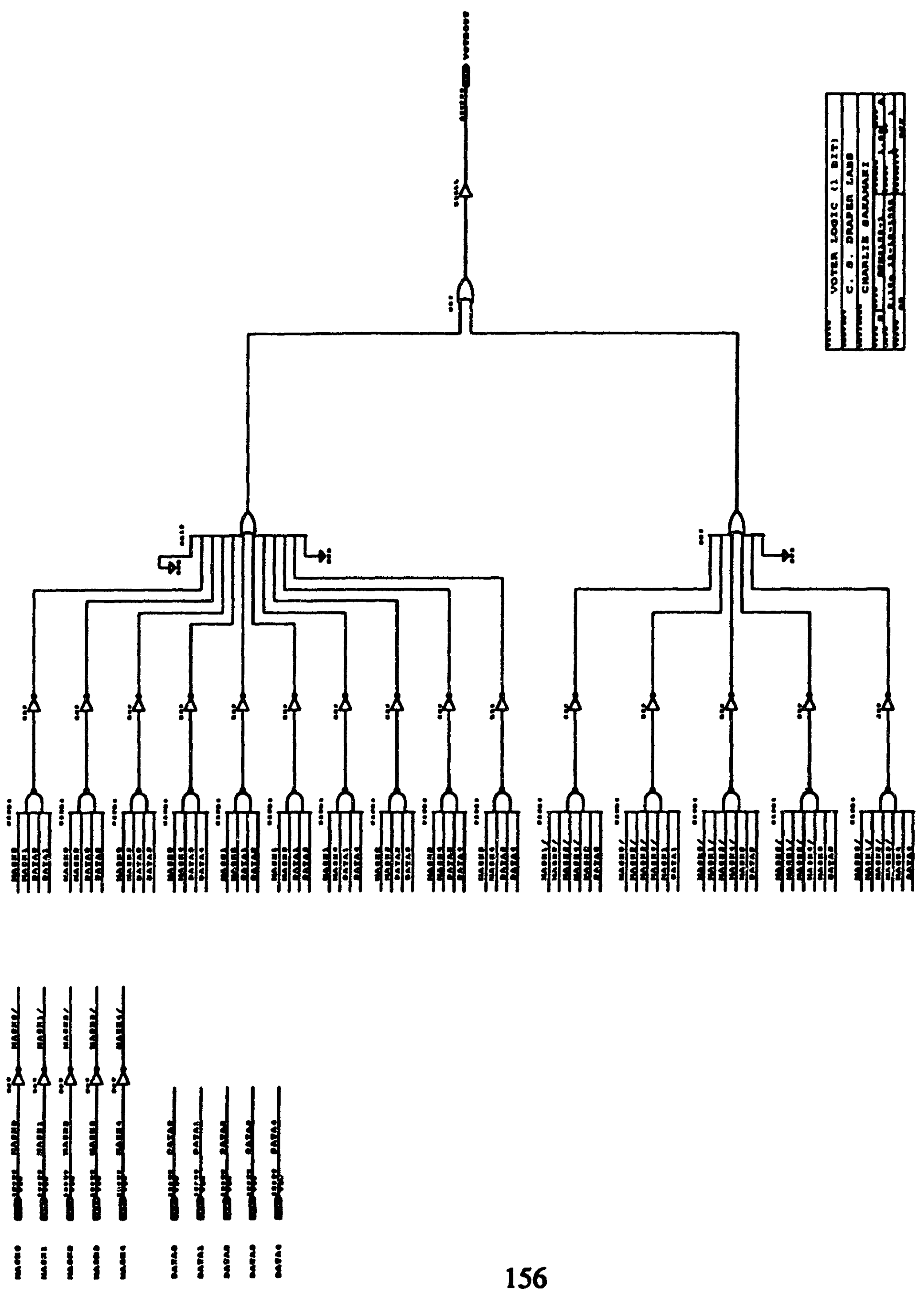



IN100
IN011
IN010
IN001
IN000
SEL0
SEL1
SEL2
INPUT
VCC
NOT
SEL0/
SEL1/
SEL2/
AND4
SEL2
SEL1/
SEL0/
SEL2/
SEL1
OR6
GND
OUTPUT
OUT
TITLE 5 TO 1 MULTIPLEXOR
COMPANY C. S. DRAPER LABS
DESIGNER CHARLIE SAKAMAKI
SIZE C
EPLD EPM5128-1
NUMBER 1.00
REV A
DATE 3:17p 9-11-1990
SHEET 1 OF 1
TURBO ON
SECURITY OFF

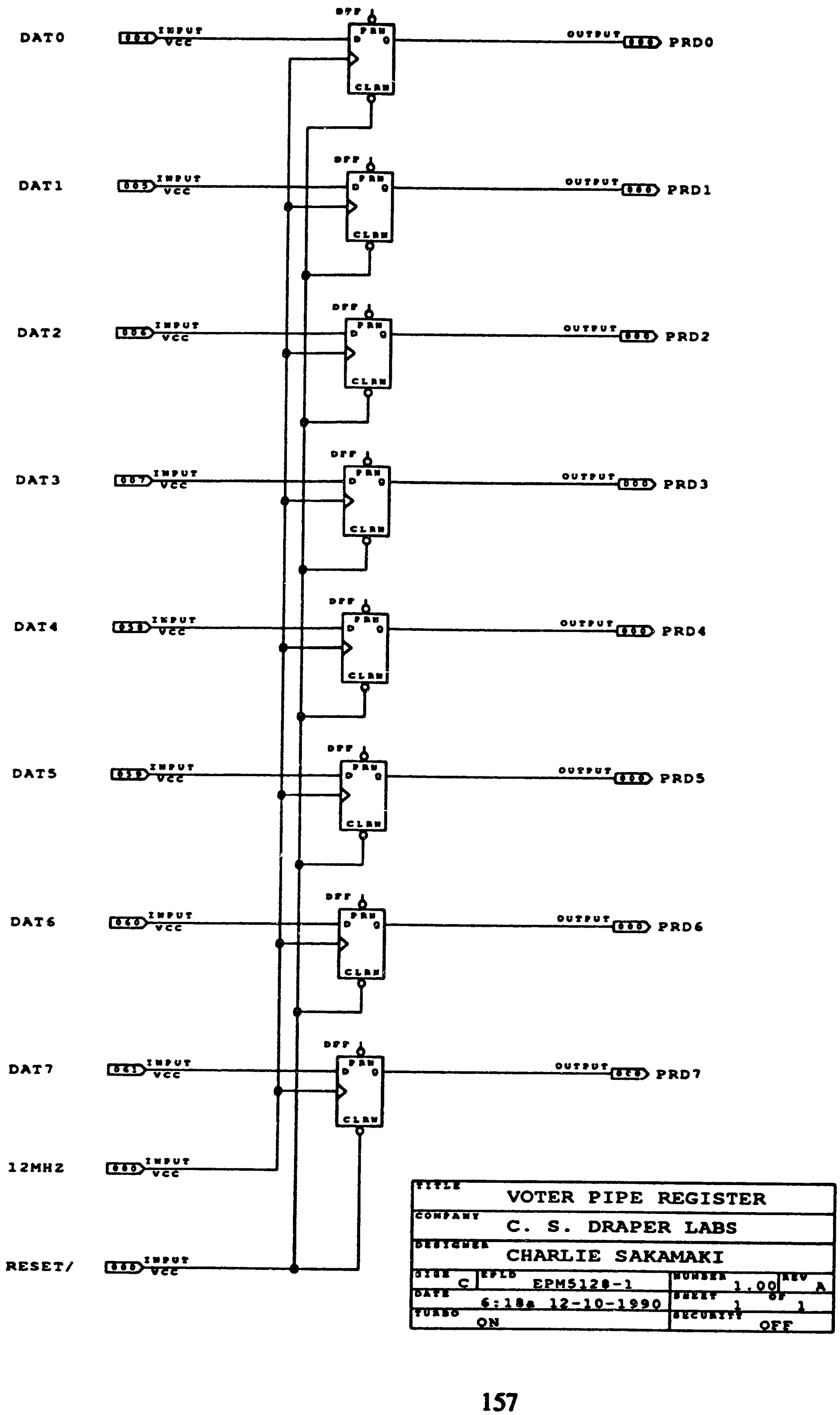
Appendix B - Voter Chip Schematics



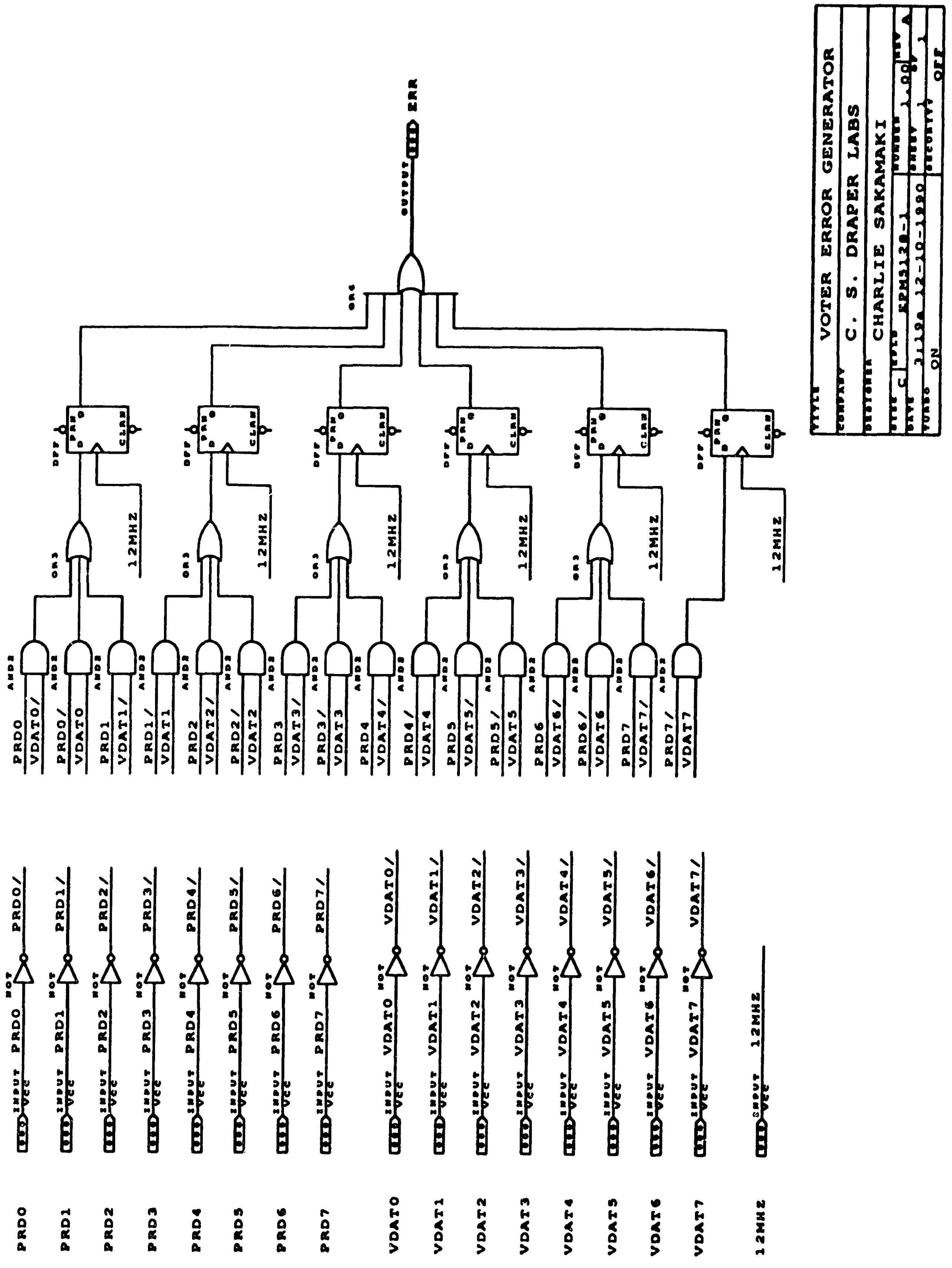
Appendix B - Voter Chip Schematics



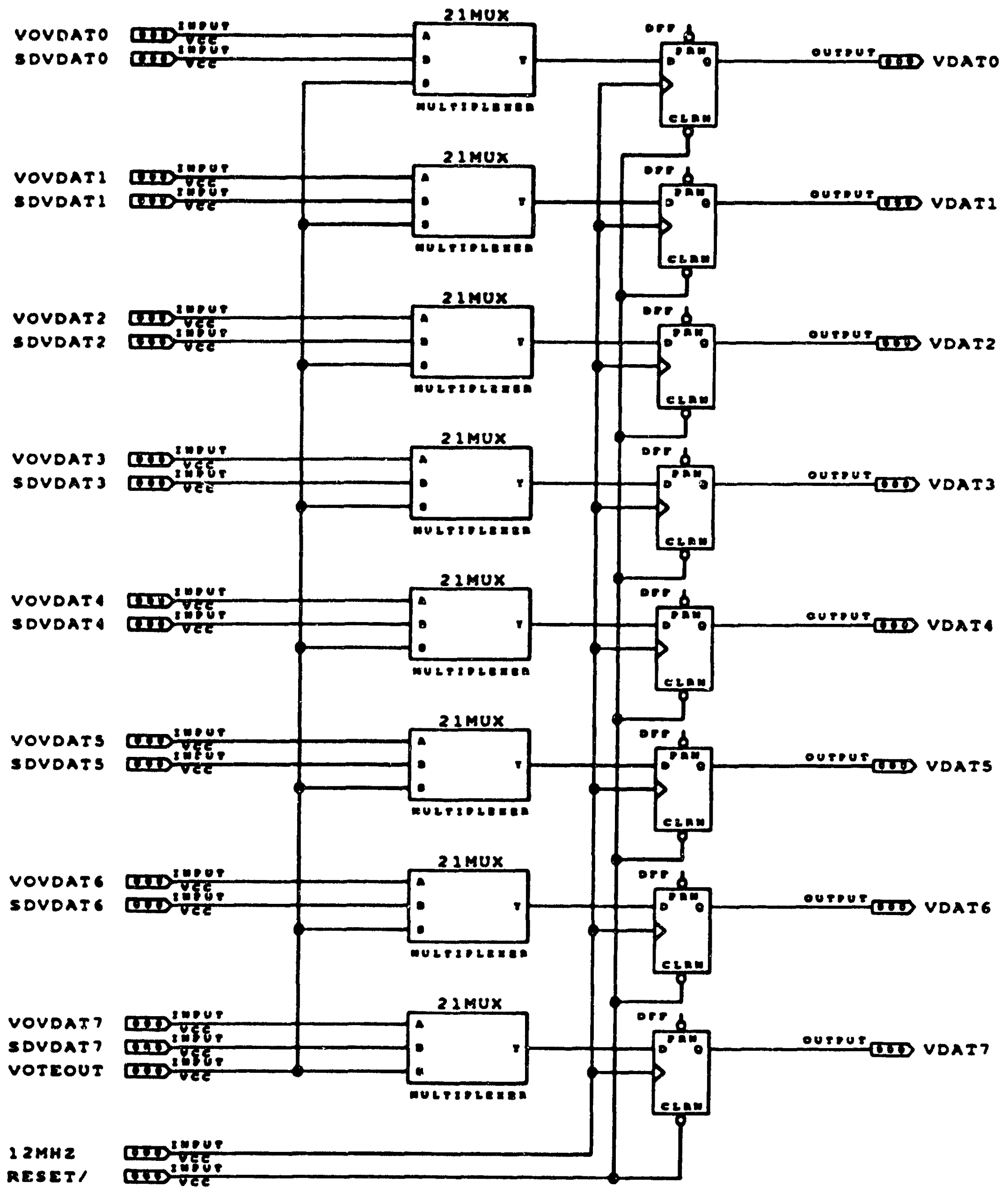
Appendix B - Voter Chip Schematics



Appendix B - Voter Chip Schematics

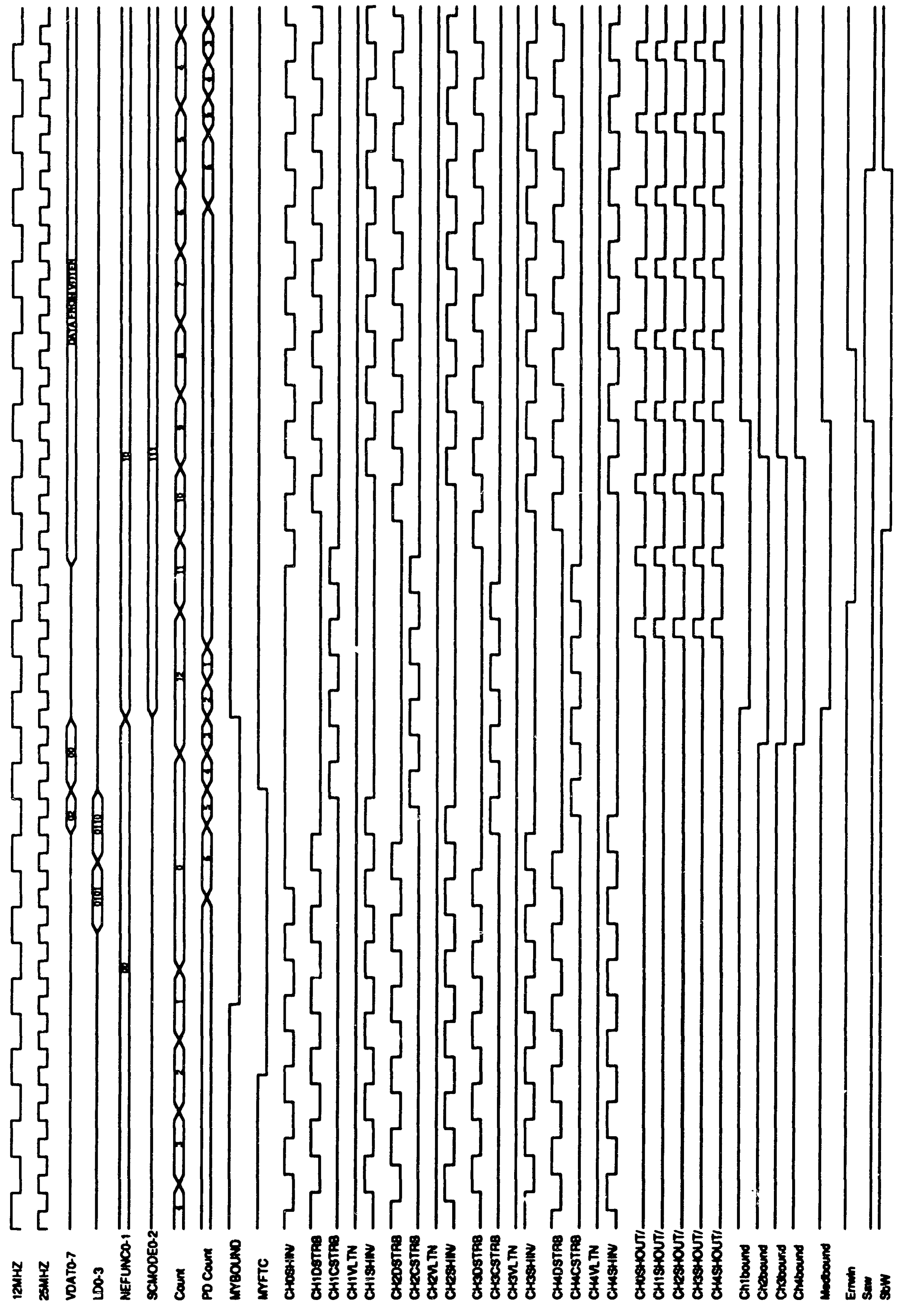


Appendix B - Voter Chip Schematics



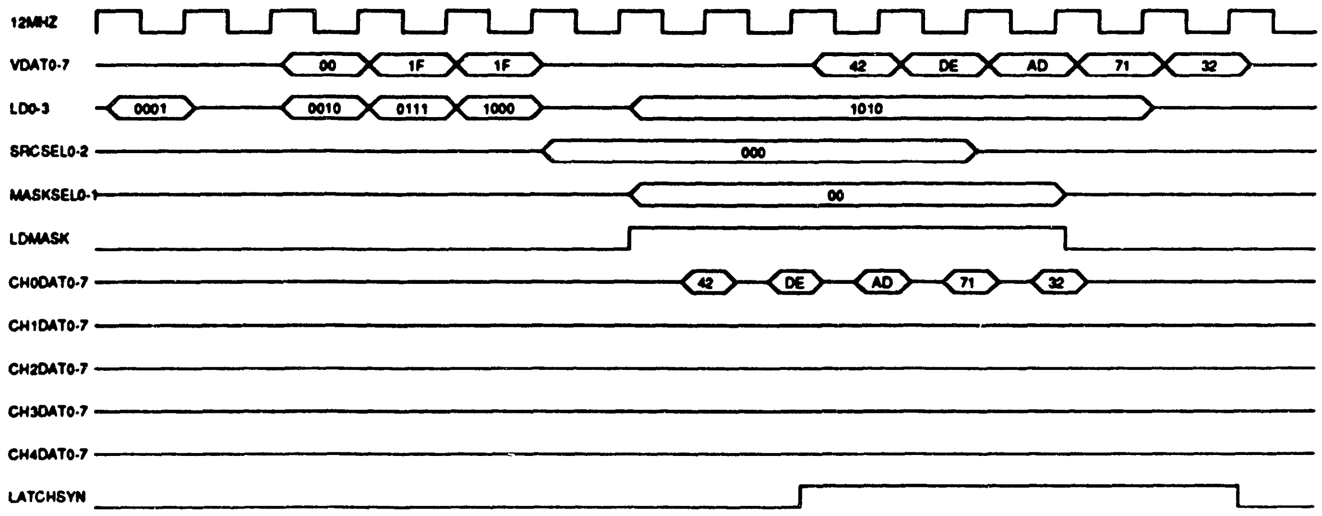
TITLE	VOTER CHIP BUS DRIVER		
COMPANY	C. S. DRAPER LABS		
DESIGNER	CHARLIE SAKAMAKI		
SIZE C	EPLD EPM5128-1	NUMBER 1.00	REV A
DATE	11:14p 12-05-1990	SHEET 1	OF 1
TURBO	ON	SECURITY	OFF

APPENDIX C - FTC Chip Simulation

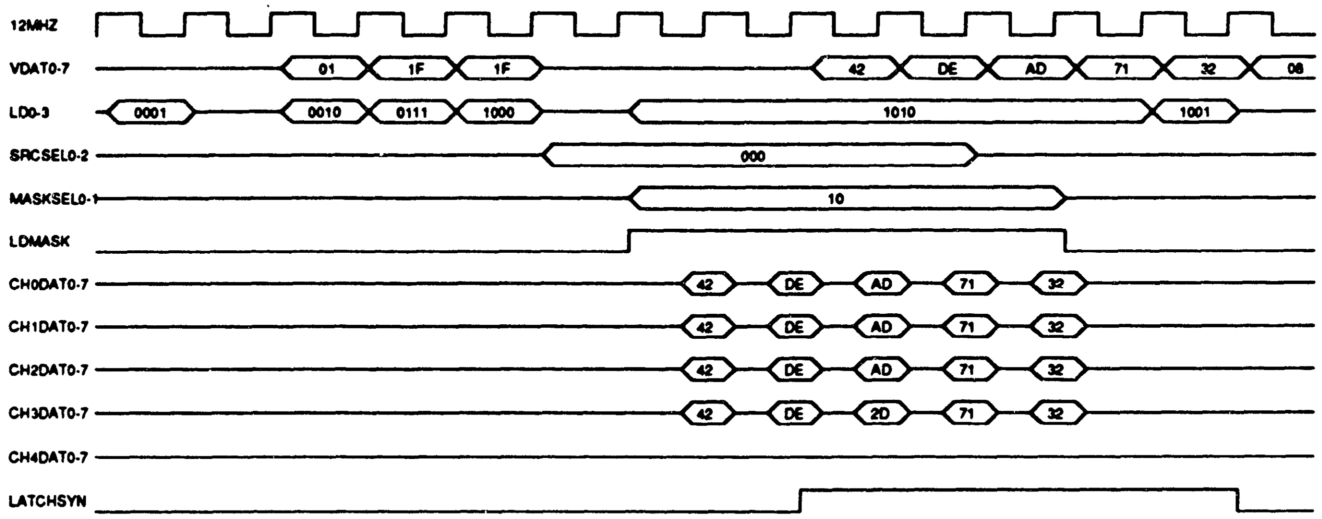


APPENDIX D - Voter Chip Simulation

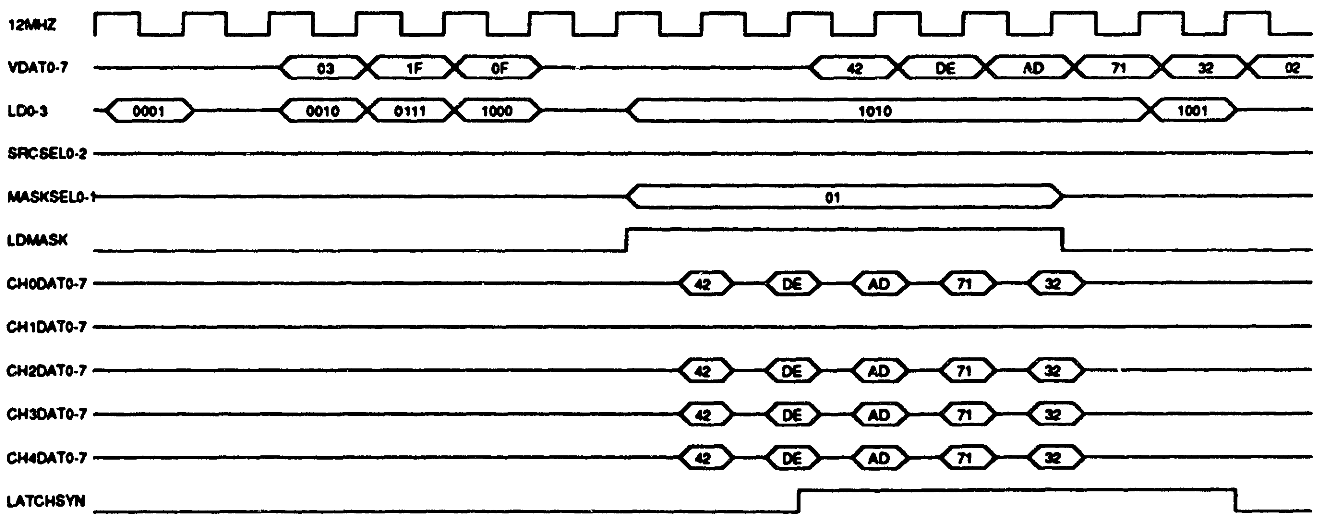
Using Reflect Masks



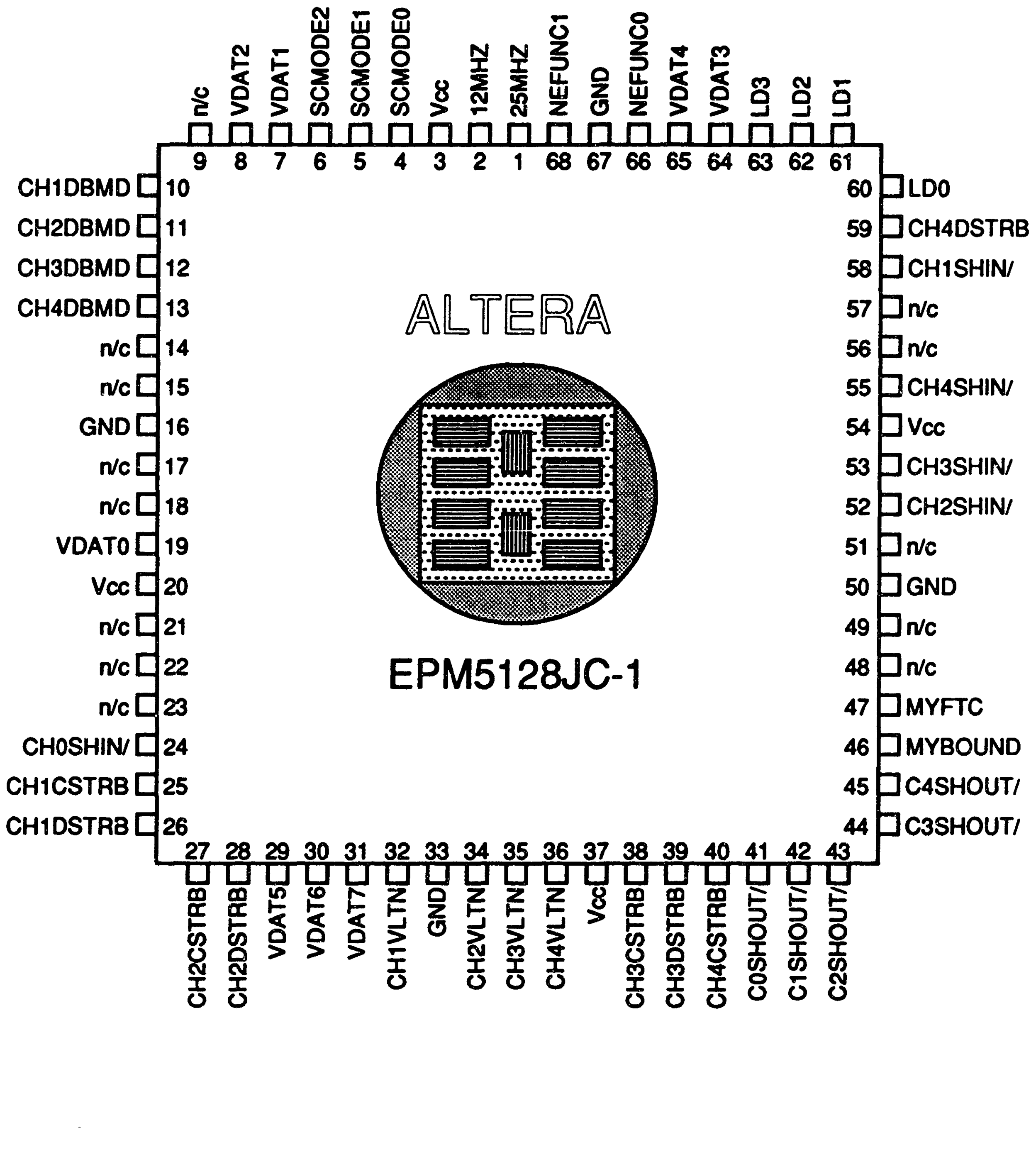
Using From Masks



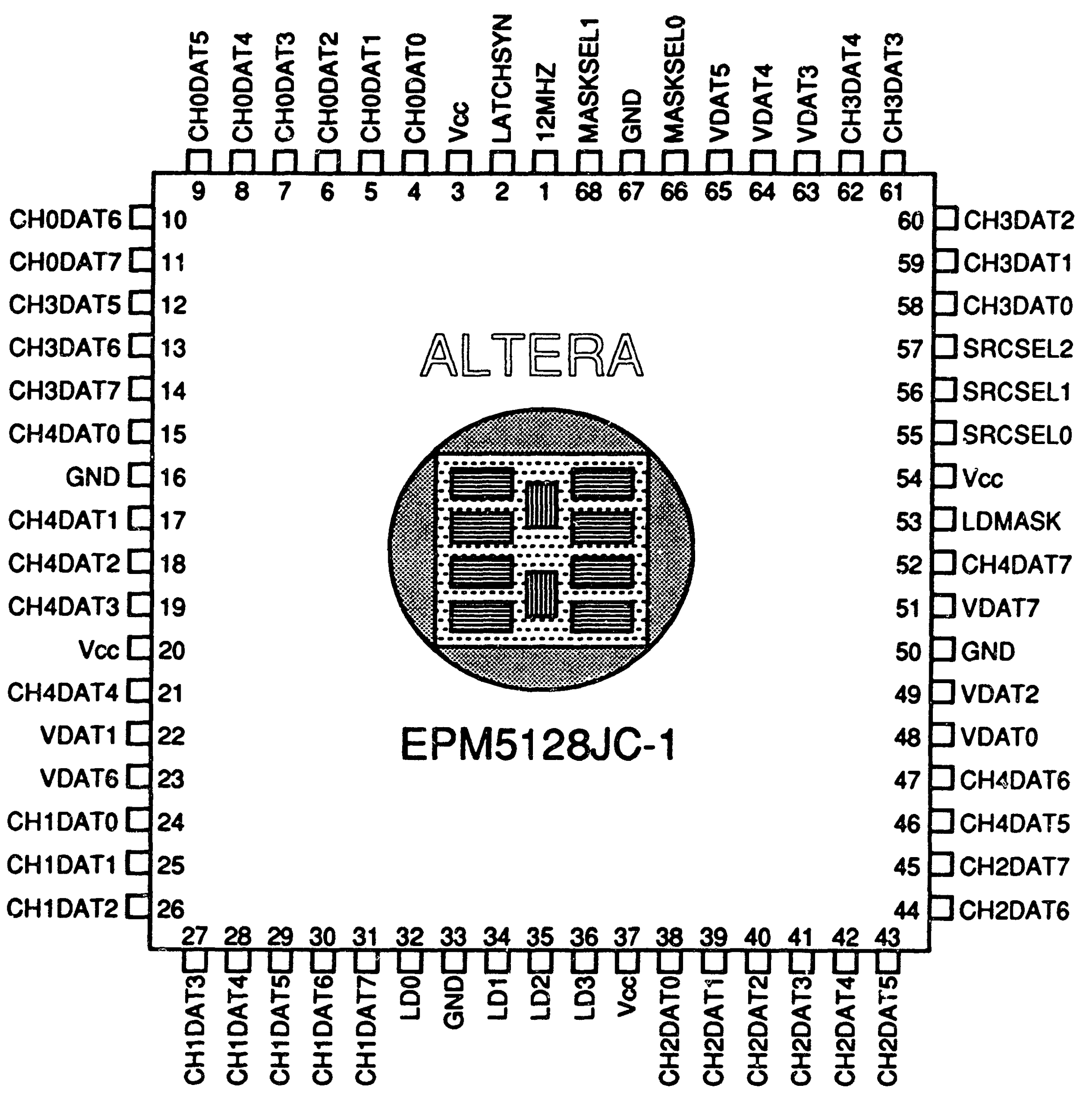
Using Vote Masks



APPENDIX E
FTC Chip Pinouts



APPENDIX F
Voter FPGA Pinouts



References

[AH78] A. Hopkins, T. B. Smith, and J. H. Lala. "FTMP--A Highly Reliable Fault-Tolerant Multiprocessor for Aircraft," *Proceedings of the IEEE*. Vol. 66, No. 10, October 1978, pp. 1221-1239.

[AMF88] A. M. Finn, R. M. Kieckhafer, P. M Thambidurai. The MAFT architecture for distributed fault tolerance. *IEEE Trans. Computers*. Vol. 37, No.4, April 1988, pp. 398-405.

[AIPS84] *Advanced Information Processing System (AIPS) System Specification*. CSDL Report, May 1984.

[ALT90] *The Maximalist Handbook*, an Altera databook, January 1990.

[DD82] D. Dolev. "The Byzantine Generals Strike Again," *Journal of Algorithms*. Vol. 3, 1982, pp. 14-30.

[DD84] D. Dolev, J. Y. Halpern, B. Simons, and H. R. Strong. "Fault-Tolerant Clock Synchronization," *Communications of ACM*, 1984, pp. 89-101.

[DD84-2] D. Dolev, J. Y. Halpern, and H. R. Strong. "On the Possibility and Impossibility of Achieving Clock Synchronization," *Communications of ACM*. 1984, pp. 504-511.

[DD84-3] D. Dolev, C. Dwork, and L. Stockmeyer. *On the Minimal Synchronism Needed for Distributed Concensus*. IBM Research Report RJ 4292(46990), IBM, May 1984.

[DG77] D. Gangsaas and D. L. Martin. "Testing of the YC-14 Flight Control System Software," *Guidance and Control*. Vol. 1, No. 4, pp. 242-247.

[DJK90] D. J. Klinger, Y. Nakada, and M. A. Menendez. *AT & T Reliability Manual*. Van Nostrand Reinhold, 1990.

[FFT87] F. F. Tsui. LSI/VLSI Testability Design. McGraw-Hill, Inc., 1987.

[INT86] *Intel Reliability Report*. Report Number RR-35E. December 1986.

[JFM75] J. F. Meyer and R. J. Sundstrom. "On-Line Diagnosis of Unrestricted Faults," IEEE Transactions on Computers. Vol. c-24, No. 5, May 1975, pp. 468-475.

[JG84] J. Goldberg. *Development and Analysis of the SIFT Computer*. NASA Contract Report 172146, SRI International, February 1984.

[JHW78] J. H. Wensley. "SIFT: Design and Analysis of a Fault-Tolerant Computer for Aircraft Control," *Proceedings of the IEEE*. Vol.66, No. 10, October 1978, pp. 1255-1268.

[JJD88] J. J. Deyst, R. E. Harper, and J. H. Lala. "Fault Tolerant Parallel Processor Architecture Overview," CSDL Report Number CSDL-P-2780, June 1988.

[KE85] K. Eshraghian and N. H. E. Weste. *Principles of CMOS VLSI Design.* AT&T Bell Laboratories, Incorporated, 1985.

[LL78] L. Lamport. "Time, Clocks, and the Ordering of Events in a Distributed System," *Communications of ACM.* Vol. 21, No. 7, July 1978.

[LL82] L. Lamport, R. Shostak, and M Pease. "The Byzantine Generals Problem," *ACM Transactions of Programming Languages and Systems.* Vol. 4, No. 3, July 1982, pp. 382-401.

[LSA86] L. S. Alger, M. J. Dzwonczyk, R. J. Gauthier, J. H. Lala. "A Fault Tolerant Processor to Meet Rigorous Failure Requirements," Unpublished CSDL Report, July 1986.

[MIL86] Reliability Prediction of Electronic Equipment. Military Handbook MIL-HDBK-217E, October 26, 1986.

[MJF82] M. J. Fischer, R. J. Fowler, and N. A. Lynch. "A Simple and Efficient Byzantine Generals Algorithm," *Proceedings of the IEEE*, 1982, pp. 46-52.

[MJF82-2] M. J. Fischer and N. A. Lynch. A Lower Bound for the Time to Assure Interactive Consistency," *Information Processing Letters.* Vol. 14, No. 4, June 1982, pp. 183-186.

[MJF85] M. J. Fischer, N. A. Lynch, and M. S. Patterson. "Impossibility of Distributed Concensus with One Faulty Process," *Journal of the ACM.* Vol. 32, No. 2, April 1985, pp. 374-382.

[MP80] M. Pease, R. Shostak, and L. Lamport. "Reaching Agreement in the Presence of Faults," *Journal of the ACM.* Vol. 27, No. 2, April 1980, pp. 229-234.

[REH85] R. E. Harper. "The FTPP Cluster," Unpublished CSDL Report, April 1985.

[REH87] R. E. Harper. *Critical Issues in Ultra-Reliable Parallel Processing.* Doctor of Philosophy Thesis, Massachusetts Institute of Technology, June 1987.

[REH87-2] R. E. Harper. "Fault Tolerant Parallel Processor Architecture Overview," Unpublished CSDL Report, November 1987.

[REH88] R. E. Harper. "Reliability Analysis of Parallel Processing Systems." CSDL Report Number CSDL-P-2853, October 1988.

[RLH87] R. L. Heyda. *A Message Passing System for A Fault Tolerant Parallel Processor.* Bachelor of Science and Master of Science Thesis, Massachusetts Institute of Technology, May 1987.

[RWB85] R. W. Butler, C. M. Krishna, K. G. Shin. "Ensuring Fault Tolerance of Phase-Locked Clocks," *IEEE Transactions on Computers*. Vol. c-34, No. 8, August 1985, pp. 752-756.

[RWB86] R. W. Butler and D. L. Palumbo. "A Performance Evaluation of the Software-Implemented Fault-Tolerance Computer," *Journal of Guidance*. Vol. 9, No. 2, March-April 1986, pp. 175-180.

[SAF87] S. A. Friend. *Process Synchronization within a Loosely Coupled Fault Tolerant Parallel Processing System*. Master of Science Thesis, Northeastern University, December 1986.

[SJA89] S. J. Adams, M. J. Dzwonczyk, R. J. Gauthier, and M. F. McKinney. *Avionics Architecture Studies for the Entry Research Vehicle*. NASA Contractor Report 181828, May 1989.

[TAA88] T. A. Abler. *A Network Element Based Fault Tolerant Processor*. Master of Science Thesis, Massachusetts Institute of Technology, May 1988.

[TBS81] T. B. Smith. "Fault-Tolerant Clocking System". A CSDL Report. 1981.