NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

DESIGNING FAST GOLAY ENCODER/DECODER IN XILINX XACT WITH MENTOR GRAPHICS CAD INTERFACE

by

Mehmet Sarı

March 1997

Thesis Advisor:	Chin Hwa Lee
Second Reader:	Todd Weatherford

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE DESIGNING FAST GOLAY ENCODER/DECODER IN XILINX XACT WITH MENTOR GRAPHICS CAD INTERFACE	5. FUNDING NUMBERS
6. AUTHOR(S) Sarý, Mehmet	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

The programmable logic array is one of the most fascinating and fast developing areas of technology. Field programmable gate arrays are becoming prevalent in design as the density of the gate arrays goes up. In this thesis study, a fast encoding/decoding algorithm, Extended Golay Coding, is implemented in Xilinx XC4000 family programmable gate array (FPGA) architecture. The encoder/decoder is designed using the Xilinx XACT tool with the Mentor Graphics schematic capture Design Architect (DA) and QuicksimII simulation interfaces. With the static RAM bits onboard the new Xilinx FPGAs, the architecture is more powerful, and it is relatively easy to upgrade the old design based on the needs of the users. In this thesis, fast encoder/decoder is implemented with transmission word redundancy and interleaving. This is based on the data link layer description of the Milstd 181-144A. The FPGA static RAM bits are used for the encode and decode ROM of the algorithm that makes the coder faster. Modular approach and design hierarchy made design tasks easier and upgradable in this study. The timing simulations of some design modules will be presented. Due to the complexity of the circuits, it is found that the design has to be migrated to a higher density chip than XC4003 used in the simulations.

14. SUBJECT TERMS Spinning Sphere, Rotationally Symmetric Flow, Spectral Methods			15. NUMBER OF PAGES 103
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

DESIGNING FAST GOLAY ENCODER/DECODER IN XILINX XACT WITH MENTOR GRAPHICS CAD INTERFACE

Mehmet Sarı
Lieutenant Junior Grade, Turkish Navy
B.S., Turkish Naval Academy, 1991

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1997

Author: ______________________________
Mehmet Sarı

Approved by: ______________________________
Chin-Hwa Lee, Thesis Advisor

____
Todd Weatherford, Second Reader

____
Herschel H. Loomis, Jr., Chairman
Department of Electrical and Computer Engineering

ABSTRACT

The programmable logic array is one of the most fascinating and fast developing areas of technology. Field programmable gate arrays are becoming prevalent in design as the density of the gate arrays goes up. In this thesis study, a fast encoding/decoding algorithm, Extended Golay Coding, is implemented in Xilinx XC4000 family programmable gate array (FPGA) architecture. The encoder/decoder is designed using the Xilinx XACT tool with the Mentor Graphics schematic capture Design Architect (DA) and QuicksimII simulation interfaces. With the static RAM bits onboard the new Xilinx FPGAs, the architecture is more powerful, and it is relatively easy to upgrade the old design based on the needs of the users. In this thesis, fast encoder/decoder is implemented with transmission word redundancy and interleaving. This is based on the data link layer description of the Milstd 181-144A. The FPGA static RAM bits are used for the encode and decode ROM of the algorithm that makes the coder faster. Modular approach and design hierarchy made design tasks easier and upgradable in this study. The timing simulations of some design modules will be presented. Due to the complexity of the circuits, it is found that the design has to be migrated to a higher density chip than XC4003 used in the simulations.

TABLE OF CONTENTS

I. INTRODUCTION

The effective performance of a station, while communicating over adverse RF channels, relies on the combined use of error correction, interleaving and redundancy. These functions shall be performed within the transmit encoder and receive decoder. By encoding and decoding redundant bits, it is possible to correct bit errors without asking the source to retransmit. This provides reliable transmission in a noisy or heavily interferenced channel. The bit errors at the receiving end are recovered by forward error correcting (FEC) codes without requiring a second channel to ask for retransmit (like in most error correcting schemes using automatic repeat request - ARQ [Snelgrove,1994]). Block codes or convolutional codes are used where retransmission of data is impractical or impossible, such as in space probes or in broadcast satellites that transmit to multiple receivers simultaneously [Stallings, 1994].

In this thesis study, the implementation of the Fast Golay encoder/decoder is done on schematic entries. The design can be exported into a Programmable Logic Device. Functional, and timing simulations of the design will be explained in this thesis.

The main objective of this thesis is to design a reliable encoder/decoder that can be used in a noisy or heavily interfered environment. The functionality of the algorithm is explained in Chapter II. The designs in schematics are explained in Chapter III. In Chapter IV, the stimulus inputs and simulation results of the schematic sheets are discussed. The controller part is to be designed according to these simulations results in a later study. The Xilinx Logical Cell Array (LCA) technology is mentioned in Chapter V. The conclusion and recommendations are given in the Chapter VI of this thesis.

II. ENCODING/DECODING AND EXTENDED GOLAY CODE

This chapter explains the basics of encoding/decoding algorithms, the value of using the extended Golay Code, and how the algorithm works.

The demand for efficient and reliable digital data communication has been increased by the emergence of large scale and high speed data networks in the military, governmental, and private sectors. A merging of communications and computer technology in this type of systems require the error correcting algorithms. They are essential to provide reliable transmission of data. [Snelgrove, 1994].

A typical transmission system may be represented by the block diagram in Figure 2.1 [Lin and Castello, 1983]. The information source can be either a person or a machine, e.g. digital computer. The source output can be either a continuous waveform or a sequence of discrete symbols. The source encoder transforms source output into a sequence of binary digits (bits), called the information sequence u in Figure 2.1. In the case of a continuous source, this involves analog-to-digital (A\D) conversion. The source encoder helps to represent a source output with minimized number of bits per unit time. Another requirement for source encoder is the ability to reconstruct signal from information sequence u without ambiguity.

The channel encoder transforms the message sequence u into a discrete encoded sequence x in Figure 2.1, called a code word. The modulator changes

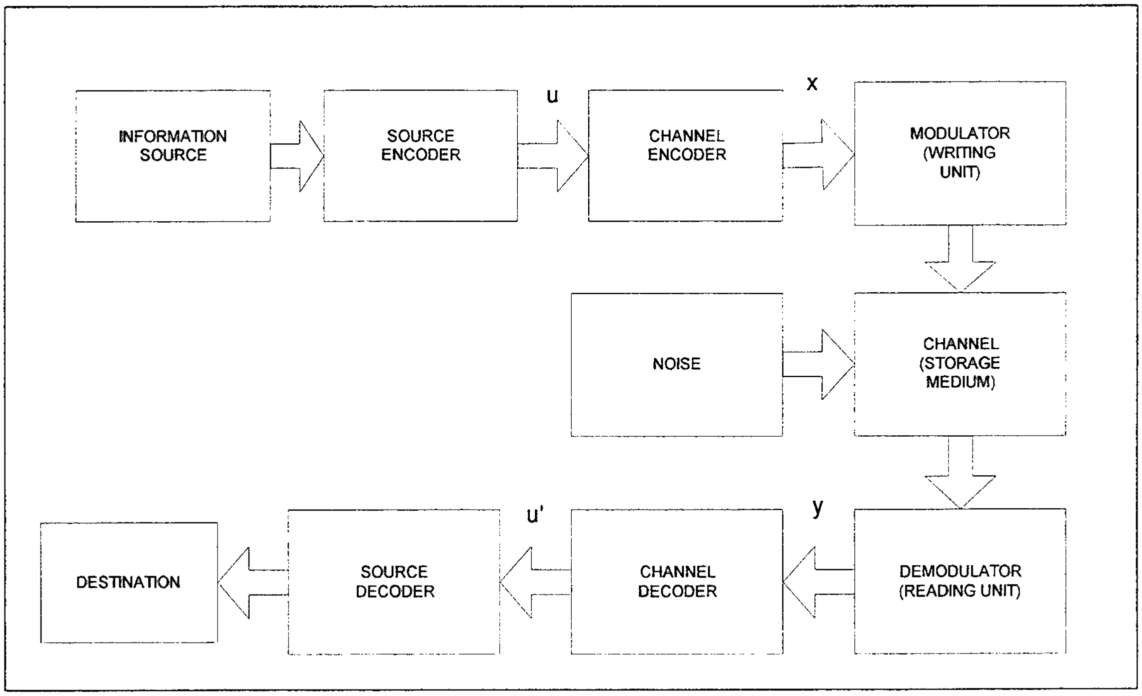

Figure 2.1. Block Diagram of a Typical Data Transmission or Storage System.

each output symbol of the channel encoder into a waveform of duration T seconds suitable for transmission. This waveform enters the channel and is corrupted by noise. The demodulator processes each received waveform of duration T and produces an output sequence y. The channel decoder transforms the received sequence y (shown below in Figure 2.1) corresponding to the encoded sequence x, into a binary sequence u'. The source decoder then transforms the estimated sequence u' into an estimate of the source output and delivers the estimate to the destination [Lin/Cas., 1983].

A. FEC IN MILSTD 188 - 141A

Milstd 188-141A is a high frequency (HF) military communication standard. The Forward Error Correcting (FEC) in this standard is the subject of study in this thesis given in this chapter. This standard consists of Automatic Link Establishment (ALE) and FEC sublayers in the Data Link Layer of the seven layer OSI model shown in Figure 2.2. This thesis involves only the Golay Code, interleave, and redundancy of the standard.

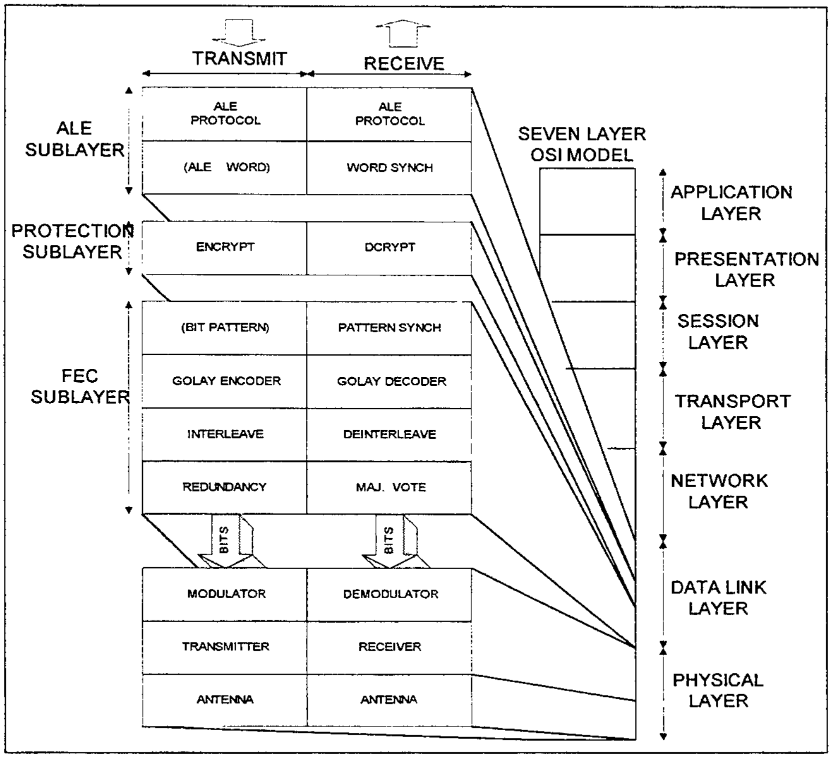

Figure 2.2. Milstd 188-141A FEC in OSI Model.

B. THE GOLAY CODE

The Golay code is a binary linear block (23, 12) code with minimum Hamming distance = 7 [Lin and Castello, 63 and Proakis, 1989, 446]. The (23, 12) Golay code is the only known multiple-error-correcting binary code which is capable of correcting any combination of three or less random errors in a block of 23 digits [Lin,Castello, 1983]. The extended Golay code is obtained by adding an overall parity bit to the (23, 12) code which results in a binary linear block (24, 12) code with minimum Hamming distance = 8.

The (23 ,12) Golay code is generated either by

$$g_1(x) = 1 + x^2 + x^4 + x^5 + x^6 + x^{10} + x^{11} \tag{2.1}$$

or by

$$g_2(x) = 1 + x + x^5 + x^6 + x^7 + x^9 + x^{11} \tag{2.2}$$

Both $g_1(x)$ and $g_2(x)$ are factors of $x^{23} + 1$ and $x^{23} + 1 = (1 + x)^* g_1(x)^* g_2(x)$ [Lin and Castello, 135]. The (24, 12, 3) Golay FEC (Forward Error Correcting) code used in this study is derived from Equation 2.2 by adding an even parity bit for each row's 23 elements to the end of the row in the generator matrix, G, derived from $g_2(x)$. The generator matrix, G, contains an identity matrix, I_{12}, and a parity matrix, P, as shown in Figure 2.3.

$$
\begin{array}{cc}
 & \begin{array}{cc} \quad I_{12} \qquad\qquad & \qquad P \end{array} \\
G= & \left[\begin{array}{cc}
100\ 000\ 000\ 000 & 101\ 011\ 100\ 011 \\
010\ 000\ 000\ 000 & 111\ 1\ 10\ 010\ 010 \\
001\ 000\ 000\ 000 & 110\ 100\ 101\ 011 \\
000\ 100\ 000\ 000 & 110\ 001\ 110\ 110 \\
000\ 010\ 000\ 000 & 110\ 011\ 011\ 001 \\
000\ 001\ 000\ 000 & 011\ 001\ 101\ 101 \\
000\ 000\ 100\ 000 & 001\ 100\ 110\ 111 \\
000\ 000\ 010\ 000 & 101\ 101\ 111\ 000 \\
000\ 000\ 001\ 000 & 010\ 110\ 111\ 100 \\
000\ 000\ 000\ 100 & 001\ 011\ 011\ 110 \\
000\ 000\ 000\ 010 & 101\ 110\ 001\ 101 \\
000\ 000\ 000\ 001 & 010\ 111\ 000\ 111
\end{array}\right]
\end{array}
$$

Figure 2.3. Generator Matrix for (24, 12) Extended Golay Code.

Encoding shall use the fundamental formula:

$$\underline{x} = \underline{u} * G \tag{2.3}$$

where the code word $\underline{x}$ (1x24 matrix) shall be derived from the message word $\underline{u}$ (1x12 matrix) and the (12x24) generator matrix G, and " * " is matrix multiplication using modulo 2 addition. As seen in Figure 2.4, encoding is performed by using the encoder circuitry to output Golay check bits $G_1 \ldots G_{12}$ and data word bits in a sequence. This will be repeated three times to add word redundancy at the physical level of the communication link to increase reliability of the received word after a majority voting.

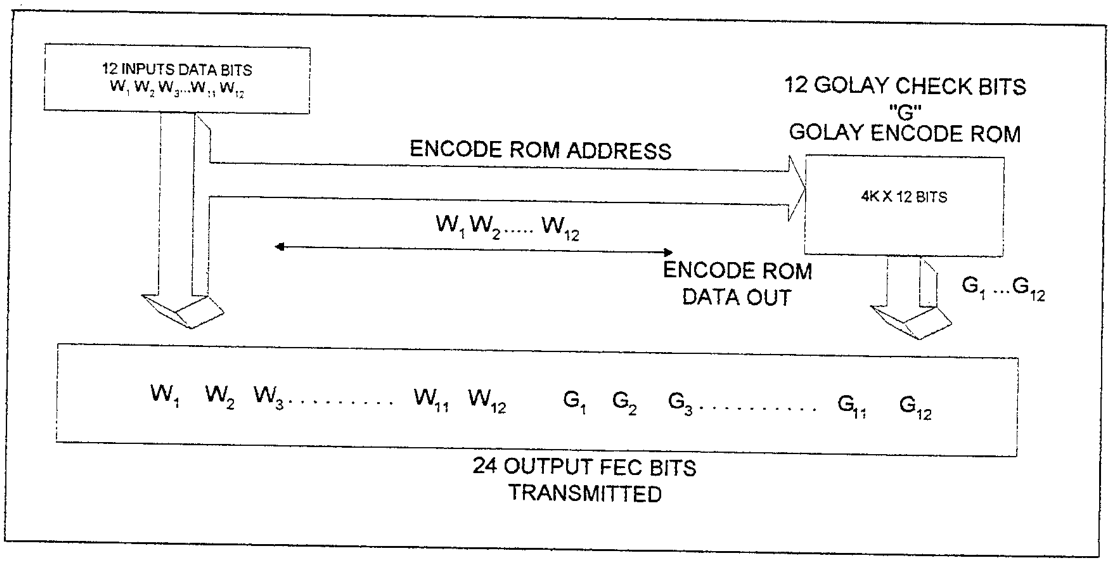

Figure 2.4. Golay FEC Encoding Example.

Decoding will implement the following equation:

$$\underline{s} = \underline{y} * H^T \tag{2.4}$$

where H^T : 24x12 matrix

$\underline{y}$: 1x24 matrix

$\underline{s}$: 1x12 matrix,

and H^T is the transpose of a parity check matrix H as shown in Figure 2.5, P^T is the transpose of the same submatrix P in generator matrix G.

$$
\begin{array}{cc}
 & \begin{array}{cc} \qquad\qquad P^T \qquad\qquad & \qquad\qquad I_{12} \qquad\qquad \end{array} \\
H = & \left[\begin{array}{cc}
111\ 110\ 010\ 010 & 100\ 000\ 000\ 000 \\
011\ 111\ 001\ 001 & 010\ 000\ 000\ 000 \\
110\ 001\ 110\ 110 & 001\ 000\ 000\ 000 \\
011\ 000\ 111\ 011 & 000\ 100\ 000\ 000 \\
110\ 010\ 001\ 111 & 000\ 010\ 000\ 000 \\
100\ 111\ 010\ 101 & 000\ 001\ 000\ 000 \\
101\ 101\ 111\ 000 & 000\ 000\ 100\ 000 \\
010\ 110\ 111\ 100 & 000\ 000\ 010\ 000 \\
001\ 011\ 011\ 110 & 000\ 000\ 001\ 000 \\
000\ 101\ 101\ 111 & 000\ 000\ 000\ 100 \\
111\ 100\ 100\ 101 & 000\ 000\ 000\ 010 \\
101\ 011\ 100\ 011 & 000\ 000\ 000\ 001
\end{array} \right]
\end{array}
$$

Figure 2.5. Parity Check Matrix for (24, 12) Extended Golay Code.

$\underline{y} = \underline{x} + \underline{e}$ is the received vector which is the modulo-2 sum of the code word $\underline{x}$ and an error vector $\underline{e}$ in the channel $\underline{s}$ in Equation 2.4 is a vector of "n-k" bits called the syndrome. Each correctable error vector $\underline{e}$ results in a unique vector $\underline{s}$. The syndrome vector $\underline{s}$ can be computed according to the Equation 2.4, and used to index a look-up table of the corresponding error vector $\underline{e}$, which is then added modulo-2 to $\underline{y}$ to yield the original code word $\underline{x}$ as shown in Figure 2.6.

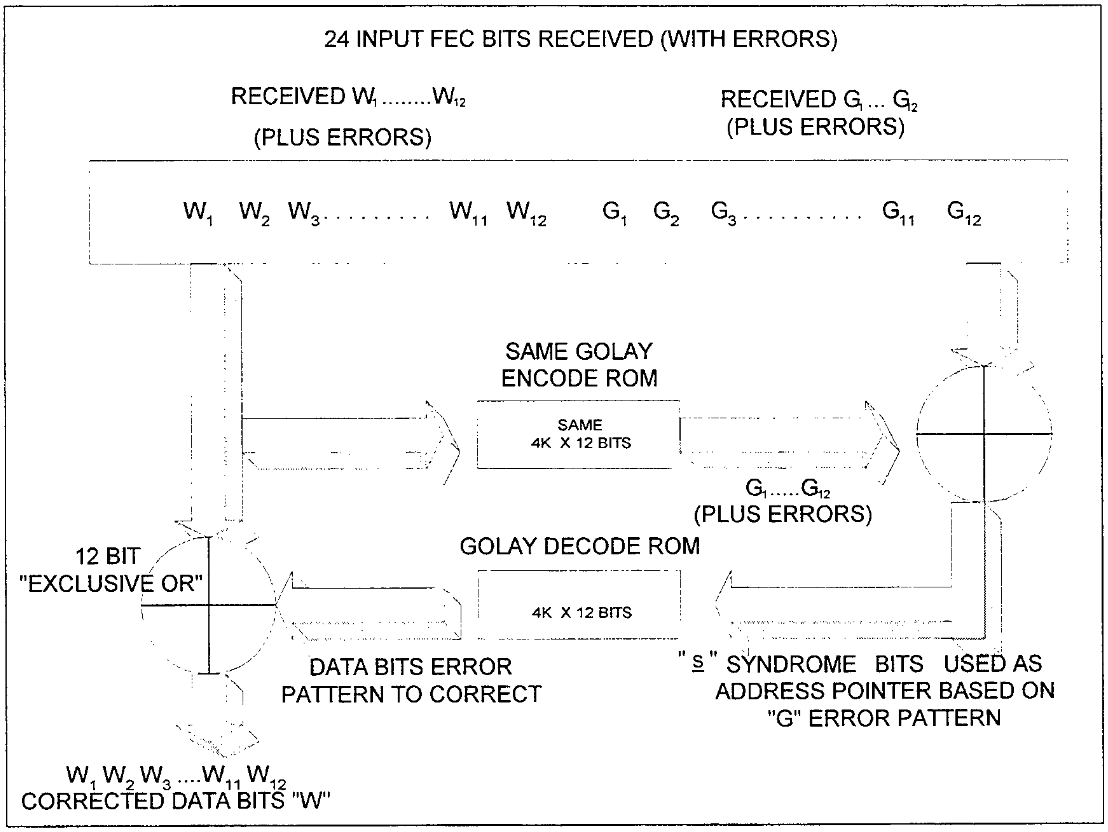

Figure 2.6. Golay FEC Decoding Example.

Flags are set according to the number of errors being corrected. If $\underline{s}$ is not equal to zero and $\underline{e}$ contains more ones than three bit errors, a detected error is indicated and appropriate flag is set.

C. INTERLEAVING AND DEINTERLEAVING

The Milstd 188-141A basic word has 24 bits, and it requires two Golay encoder section A and section B to do the work. The basic word bits Win_1 (MSB) through Win_{24} (LSB) and resultant Golay FEC bits G_1 through G_{24} (with G_{13} through G_{24} inverted) shall be interleaved following the pattern in Figure 2.7 before transmission.

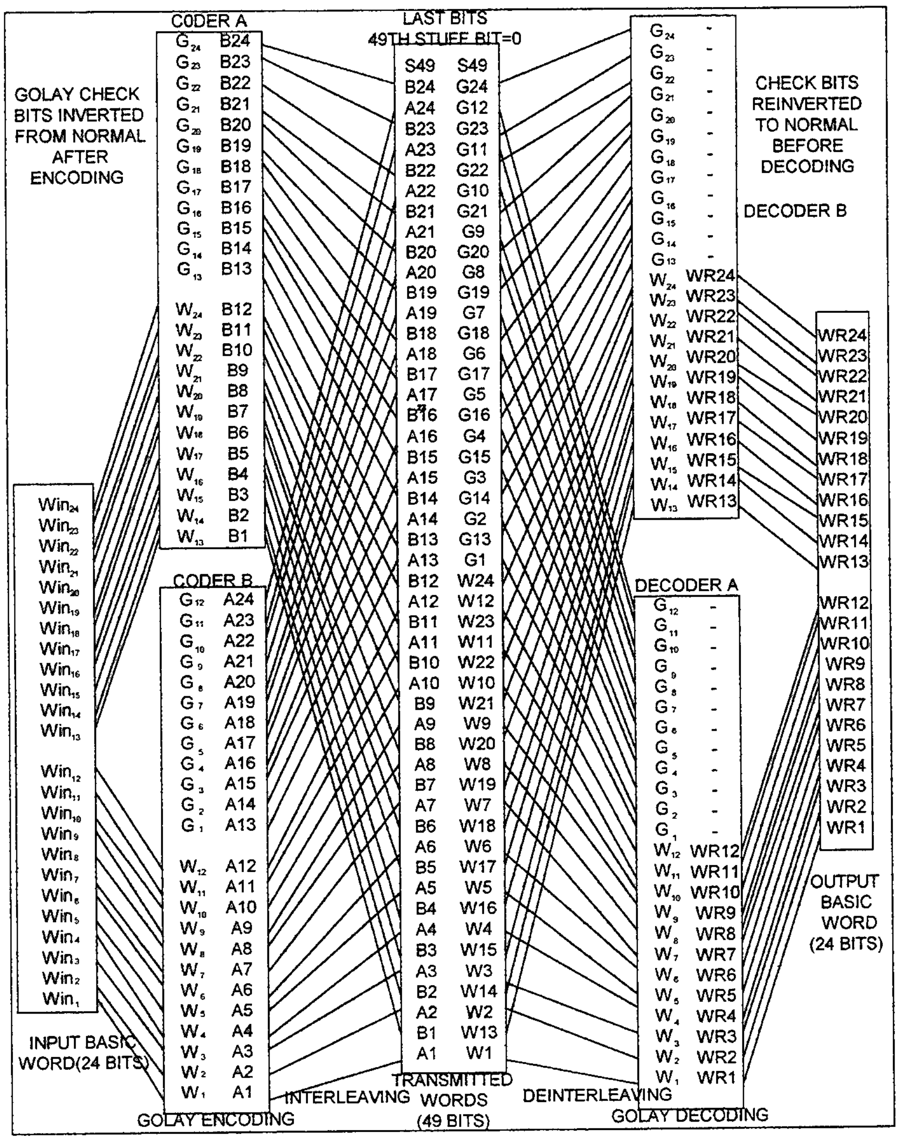

Figure 2.7. Word Bit Coding/Decoding, Interleaving/Deinterleaving.

There are forty eight interleaved bits, and the 49th bit is a stuff bit (value=0) during transmission. They shall constitute a transmitted word and be transmitted A1, B1, A2, B2, ..., A24, B24, S49 using a 16-1/3 symbols (tones) per word time (Tw). In order to lessen the effects of the channel noise causing information data bits loss all from one word, interleaving in this manner is employed. At the receiver, after 2/3 majority voting in Figure 2.8, the 48 received bits of the majority word (including the remaining errors) shall be deinterleaved as shown in Figure 2.6. The Golay FEC decoder will then produce a correct (ed) 24 bit basic word (or an uncorrectable error flag). The 49th stuff bit (S49) is ignored.

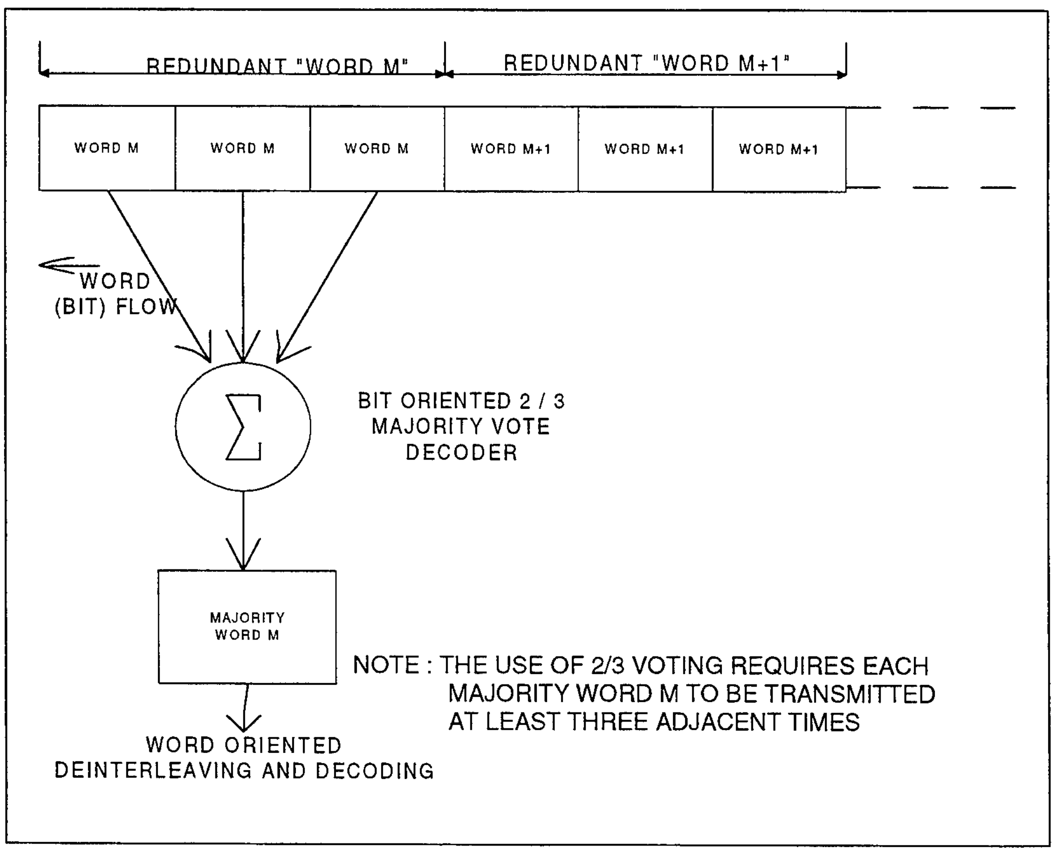

Figure 2.8. Bit and Word Majority Voting Before Decoding.

D. REDUNDANT WORDS

Each of the transmitted 49-bit (or 16-1/3 symbol) word in the Tw time shall be redundant (times 3) to reduce the effects of fading, interference and noise. At bit time interval (approximately Tw/49), the receiver shall examine the present bit and the past bit stream and perform a 2/3 majority vote, on a bit-by-bit basis, over a span of three words. The resultant 48 most recent majority bits (excluding the 49^{th} bit which is the stuff bit) constitute the latest majority word and shall be delivered to the deinterleaver and FEC decoder (Figure 2.8). In addition, the number of unanimous votes of the 48 possible votes associated with the majority word can be temporarily retained for use.

In this thesis the main focus is on the design and simulation of various modules that will accomplished the operation discussed in this chapter.

III. DESIGNING A FAST GOLAY ENCODER / DECODER

In this chapter, the implementation of the fast Golay decoder/encoder is explained. The implementation overview, and details are given to help the design of a controller in a future study.

A. MODULAR DESIGN APPROACH

The modular approach allows a designer to create small pieces of the whole design and debug them relatively easier than the bigger complex designs. Making designs in short time period is important in today's fast developing area of technology. It is advantageous to put the pieces together creating the whole design as layers of encapsulation and modules. This approach develops a hierarchy of layers from the topmost one showing the simple inputs and outputs of the whole system to the lowest one showing the gate level construction of each component. As we go down to lower layers, we get into those encapsulated design modules and involve more the hardware details of each module. In this way, the design is partitioned into layers of hierarchy and modules independent from each other. This allows them to be individually simulated and debugged.

B. GENERAL OVERVIEW OF THE DESIGN

1. Top Level Layer

a. Introduction

The top layer consists of the input signal: serial data input, encoder/decoder select, system clock, and clear signals and encode/decode ROM matrices. The output signals are encoded word transmission or decoded-and-recovered data word serial output. There are status signals representing the valid times of the signal transmission beginning and ending.

The functional overview of the design is shown in Figure 3.1. The encode, and decode ROM matrices in the algorithm are entered into the static RAM cells of the FPGA initially. The encode control signals and decode control signals are generated alongside serial input according to the encoder/decoder select input of the controller unit in Figure 3.1.

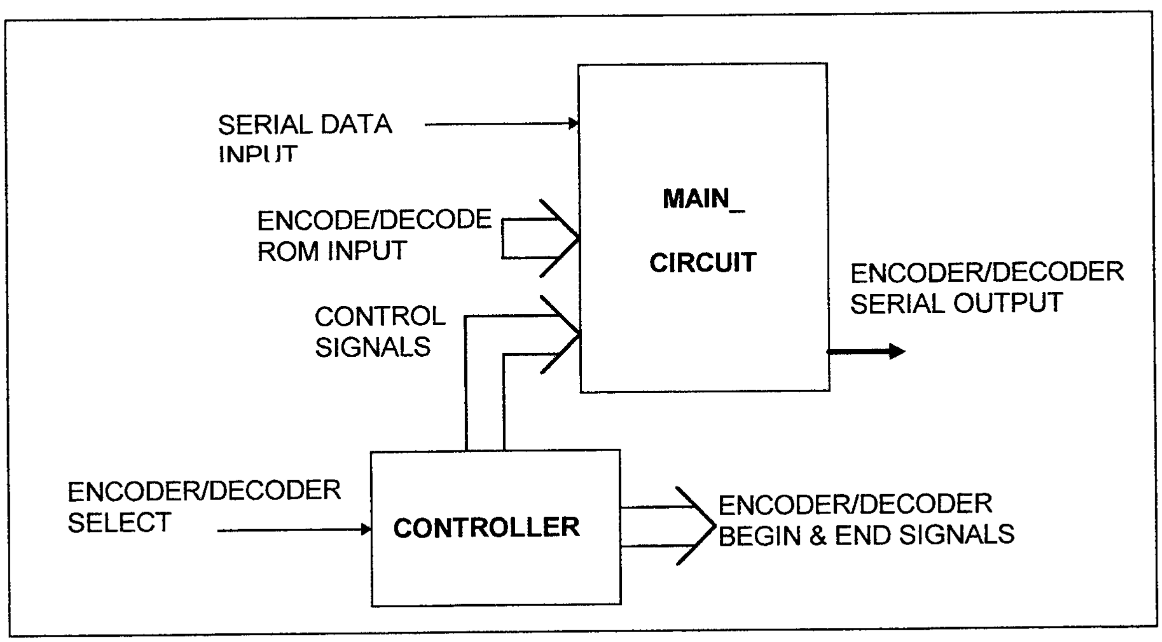

Figure 3.1. Functional Diagram of the Golay Encoder/Decoder.

The controller in Figure 3.1 will generate the status signals showing the beginning, and end of encode and decode word transmissions. The design schematic (MAIN_CIRCUIT) will generate the serial encoder/decoder output according to control signals, and serial bits coming in. The work presented in this thesis is focused on the MAIN_CIRCUIT. The design of CONTROLLER is not included.

The block diagram of the MAIN_CIRCUIT of the encoder/decoder is shown in Figure 3.2. This represents the interfaces among modules which perform the two functions shown in Figure 2.4 and Figure 2.6.

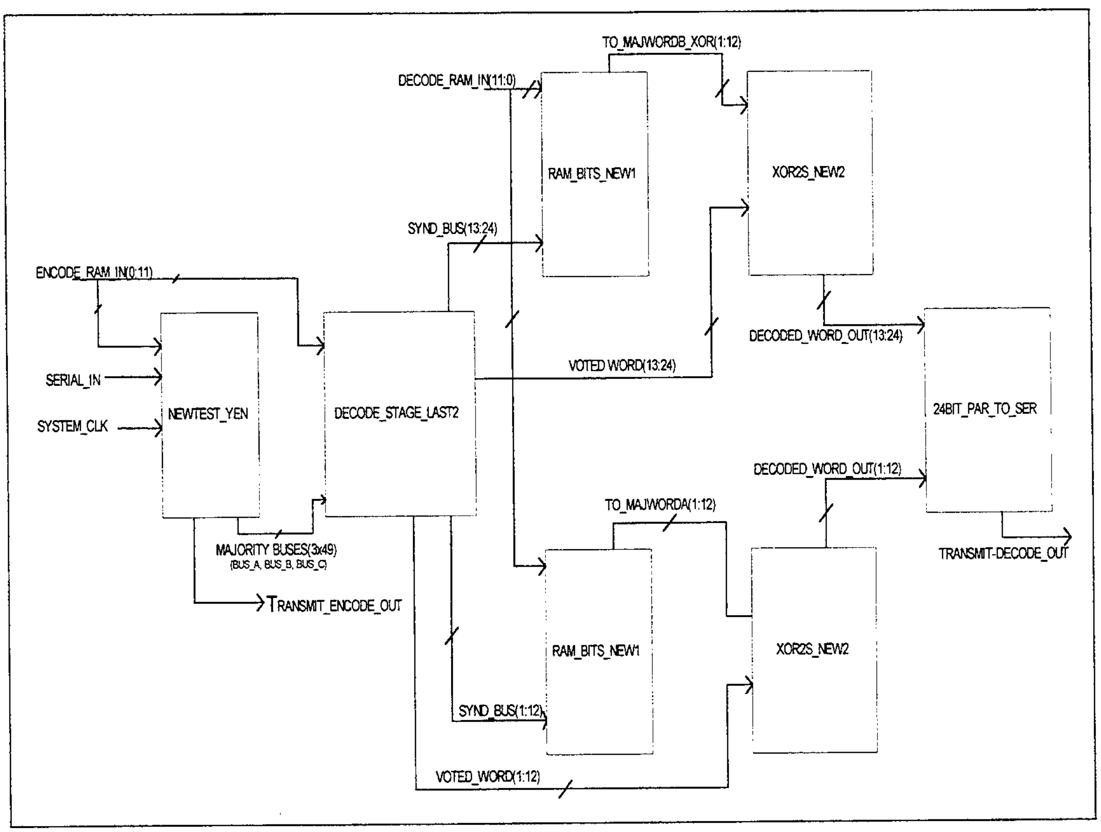

Figure 3.2 Block Diagram of the Golay Encoder/Decoder.

NEWTEST_YEN, the first module on the left of the Figure 3.2, has two functions. The first one is that the entire encoding cycle is finished in this module. The second one is that it has the three 49-bit registers that are used both for encoding, and decoding cycles as explained in details later in this chapter. The encode ROM matrix in this module is looked up by serial input message bits. The Golay check bits are generated according to the algorithm shown in Figure 2.4.

DECODE_STAGE_LAST2 is the second module on the left in Figure 3.2. One of its two basic functions is to take the three buses with received redundant message words from the first module NEWTEST_YEN to make a 2/3 majority voting explained in Chapter II. Its other function is to generate syndrome bits shown in Figure 2.6, and pass it to the RAM_BITS_NEW1 modules shown in Figure A.1. Meanwhile this module is to keep the two voted message words until they are XORed with error correcting data bits to give the corrected data word as shown in Figure 2.6.

There are two RAM_BITS_NEW1 modules. One is at the top, and the other is at the bottom. RAM_BITS_NEW1 module has the parity check matrix or decode ROM. This module takes the syndrome bits, and generates the "data bits error pattern to correct" as shown in Figure 2.6.

As shown in Figure 3.2, two identical stages are dealing with the two halves of the 12 bit portions in the same manner. XOR2S_NEW2 module takes both the received message word and "data bits error pattern bits"

TO_MAJWORDA_XOR or TO_MAJWORDB_XOR. It also corrects the received message word VOTED_WORD by XORing both inputs.

The 24BIT_PAR_TO_SER module multiplexes both corrected message words, and puts them on a serial output line.

b. Getting Into MGC Designs

The top level module Figure A.1 shows the detailed inputs, and outputs in the system. The controller generates the control signals needed for the modules. These signals will be asserted according to the programmed stimulus in QuicksimII during simulations.

The lower layer of MAIN_CIRCUIT in the hierarchy shows the modules and the interfaces among them as shown in Figure A.2. Detailed descriptions of each module will be given in this chapter.

2. Descriptions of the Design Modules

a. NEWTEST_YEN

This module has two basic functions. If the cycle is an encoding cycle, it uses serial input bits to look up the Generator matrix (G) to generate Golay check bits. The upper golay check bits are inverted before interleaving. The two Golay words and two message words are interleaved as shown in Figure A.3. This provides reliable transmission and receive of messages without being much affected by environmental noise and electromagnetic interference.

(1) ENCODE_IN_WMAT. This is the third module from the top at the left. This module's basic function is using serial input bits to look

up generator matrix for check bits. The encode and decode ROMs' values are entered into static RAM bits (ram 16x8 and ram 16x4 library parts) as shown in Figure A.4. A 4-bit counter is used to begin counting when reset. The count will be ANDed with the serial input. This will be used to look up Generator Matrix as the row select. Shift enable signal from controller will be asserted for twelve clocks. This lasts until all the message word bits to be ANDed with the count are shifted. The serial input bits will be multiplexed into the two encoders while they are also shifted into two 12 bit shift registers(12BIT_SHIFTREG1s on the left of Figure A.3) to keep the message words to be interleaved later. The contents of the shift registers and two Golay check words will be interleaved into the module REGFIRST_WEN1, the third module from right as shown in Figure A.3. The encode ROM look-up will be performed by the "count value" corresponding to ones in the message word bit sequence. The corresponding row in encode ROM will be XORed with the next one that is looked up. For example, if the eighth element of the message word is one, it will pass the count value eight, and it will look up the eighth row in the generator matrix. This row will be XORed with the present XOR module inputs. If the serial message input bit is "1", it passes the counter's count, or if it is "0", it passes "0" to look up the generator matrix. The passed count zero value will look up the first row of the matrix which consists of all zero values in its 12 bit positions and send twelve zero values into the XORing module (RAM_XOR_TO_LAST1) shown in Figure A.4. The zero bit values in the message word will not affect the result.

The 12FDC_WES and the 5FDC_WES are the same schematics except that in 12FDC_WES, there are 12 D flip-flops instead of 5 in 5FDC_WES. They are used as parallel shifter of the incoming bits with an enable signal as shown in Figure A.4 and Figure A.7. These two help pipeline the stages, and eliminate the possible invalid output transition into the XORing module (RAM_XOR_TO_LAST1).

The RAM_XOR_TO_LAST1 module gets the rows looked up in the encode ROM, and uses the serial message bits as enable to take the looked up row into the XORing module as shown in Figure A.6. The ANDing of serial input and inverted WE (write enable signal to write the encode or decode ROM into the static memory cells) provides this enable signal.

(2). 12BIT_SHIFTREG1. The 12BIT_SHIFTREG1 module shown in Figure A.5 is used to buffer the input message word bits and keep them as they go into the ENCODE_IN_WMAT module simultaneously. When the message bits are all taken into ENCODE_IN_WMAT module, this module will have all the twelve bits of the message word to be interleaved later.

(3). REGFIRST_WEN1. The two module outputs mentioned above are interleaved and shifted into the 49 bit register by asserting the enable signal FROM_ENCS from the controller in the encoding cycle as shown in Figure A.3, and Figure A.8. After these bits are interleaved and shifted into this module, the TR_ENC_OUT (transmit encoded word serial out signal from the controller) is asserted high for 3x49 clock cycles to transmit the interleaved 49 bits.

In the decoding cycle, SERCONT_SIG_0 and SERCONT_SIG_1 (control signals for multiplexing serial input both in the decode and the encode cycles) are both asserted high by the controller, and the serial input is passed into REGFIRST_WEN1's 49^{th} bit. The serial input bits to be decoded are shifted into this module, and later into the next two modules (49 bit REG_NEWSECOND1 register modules) in a sequential order. The 3x49 bits are taken into three modules to be loaded on three buses for the majority vote.

In the encoding cycle, the interleaved 49 bits are transmitted to physical medium by asserting TR_EN1 signal high as they are shifted down the first 49-bit register module (REGFIRST_WEN1, the third module from the right in Figure A.3). TR_EN2 control signal is asserted high to transmit the bits coming from the REGFIRST_WEN1 module into module REG_NEWSECOND1 (the second one from the right in Figure A.3). Finally TR_EN3 is asserted high to transmit the bits coming from the first REG_NEWSECOND1 module (the second module from the right in Figure A.3) into the second module REG_NEWSECOND1 (the first one from the right in Figure A.3).

(4). REG_NEWSECOND1. This module is used in two adjacent places at right as shown in Figure A.3. This module is also used in both the decoding and the encoding cycles as mentioned early in this chapter.

In the encoding cycle, after the interleaving, the module REGFIRST_WEN1's content is shifted into the first, and then into the second REG_NEWSECOND1 modules as shown in Figure A.3. While the bits are shifted into the next bit position, they are transmitted by the output signal "TR_SER_ENCOUT" as shown in Figure A.3.

In the decoding cycle, the control signals SERCONT_SIG_0, and SERCONT_SIG_1 from controller will be asserted to multiplex the serial input into the REGFIRST_WEN1 module. And they are shifted into the second and the third modules (two REGNEW_SECOND1 modules as shown in Figure A.3) until all the 3x49 bits are taken into three of the register sets.

A majority vote done MAJ_VOTE_DONE control signal will be asserted to clear the registers after the majority vote of three buses as shown in Figure A.3. The encoded word is transmitted to the output signal TR_SER_ENCOUT in Figure A.3.

b. DECODE_STAGE_LAST2

The DECODE_STAGE_LAST2 module shown in Figure A.10 performs the majority vote of the three incoming 49-bit buses from the registers explained in NEWTEST_YEN module. When all the buses are ready, the majority vote enable signal, MAJ_VOTE_EN from controller, is asserted high to make 2/3 majority vote of the bus values. After the vote, the voted 49bits are deinterleaved as shown in Figure A.10. The 2x12 bit message words and 2x12 bit Golay check words are obtained after the deinterleaving. The upper Golay check bits shown in Figure A.10 are reinverted before being taken into the upper XOR2S_NEW1 module at the right shown in the same figure. The control signal SHIFTAFMAJ_EN is asserted high after the vote. It shifts deinterleaved values into the golay (GOLAY_MAJ_WORD_A, and GOLAY_MAJ_WORD_B) and voted message word (GOLAY_MAJWORD_A, and GOLAY_MAJWORD_B) modules shown in the Figure A.10. SHIFT_INTO_ENCS (shift voted message words into the encoders control signal) shifts "the received message word with errors" into ENCODE_IN_WMAT module to look up the encode ROM. This yields the check bits to compare with the received Golay check bits to find out if any received message bit is in error. The ENCODING_DONE signal will be asserted high when all the received message word is shifted, and XORed with the received Golay bits. The REGCLR_ASYND_XOR (register clear) signal will be asserted high after the voted message word's XORing with syndrome bits as shown in Figure A.2.

(1). VOTER6. This module performs a majority vote after having all three incoming 49-bit buses (three redundant received 49 bits to be decoded) are ready. A majority vote enable signal from controller is asserted high when all buses are ready (the MAJ_VOTE_EN signal in Figure A.11). The majority vote algorithm is 2/3 voting. If the two or three bits are low (or zeros), then the vote is zero, if two or three bits are high (or ones), then the vote is one.

Suppose the incoming three bit signal names are A, B, and C, then the vote is implemented by $MAJ\_VOTE = \overline{\overline{(AxB)}x\overline{(AxC)}x\overline{(BxC)}}$ as shown in Figure A.11.

(2). GOLAY_MAJ_WORD_A and GOLAY_MAJ_WORD_B. These components have the values of the received check bits after majority vote and deinterleaving as shown in Figure A.10. The difference between these two modules is that the one above (GOLAY_MAJ_WORD_B) has all the incoming bits inverted as explained in Chapter II (Figure 2.7). The input/output bits are named according to Figure 2.7.

The deinterleaved bits coming into these modules are shifted by SHIFTAFMAJ_EN signal by asserting MAJ_VOTE_EN input pad of the each module shown in Figures A.10 and A.12. The bit values of each received message word will be kept by asserting MAJ_VOTE_EN low to allow the Q outputs of the D flip-flops circulate to keep the value they have until they are cleared as shown in Figure A.12.

(3). MAJ_WORD_A_TO_ENC and MAJ_WORD_B_TO_ENC. The function of these modules is to shift the voted message word to look up the encode ROM to find the check bits. These check bits are XORed with the received Golay check bits to give the syndrome bits shown in the Figure A.10. The interleaved bits coming into modules MAJ_WORD_B_TO_ENC, and MAJ_WORD_A_TO_ENC are shifted serially into higher and lower encoder modules (ENCODE_IN_WMAT modules shown as two modules one on top of the other at the second column from the right in Figure A.10). SHIFTIN_ENCA and SHIFTIN_ENCB input signals are asserted hi for 12 clock period for each module to shift all twelve-bit voted message word into encoders. After all the message bits are shifted, the SHIFT_EN control signal shown in Figure A.10 will be asserted low to end the shift. As long as the SHIFT_EN signal is high, it won't let the message bits (taken once into these modules) get into shifted sequence as shown in Figure A.13. This is taken care of by "and2b1" (two input AND gate with one input inverted) gates by having the SHIFT_EN signal coming into their inverted inputs. The modules are cleared by the ENCS_DONE (encode operation is done) signal.

(4). REG_FORMAJ_WORD. The REG_FORMAJ_WORD module keeps the received message word until the syndrome bits look up the decode ROM to give the "data bits error pattern to correct" shown in Figure2.6. After this error correcting pattern is obtained, it is used to correct the received message word as explained later in this chapter. This module is the same as the GOLAY_MAJWORD_A module except the naming of the input and output signals.

(5). XOR2S_NEW3. This module performs modulo two addition of each row of the encode ROM looked up by the received message word bits. This is done by XORing the columns of each looked up row in encode ROM as they come into XOR gates shown in Figure A.15. The GEN_SYNDROME control signal will shift the final value of the XORs' outputs into the D flip-flops to give the "syndrome bits" that are explained later in this chapter.

c. RAM_BITS_NEW1

This module looks similar to ENCODE_IN_WMAT module. The difference is that, in this module there is an additional sub module at the left (SYND_LAST_CIR). The RAM bits in this module are programmed to keep "decode ROM" values shown in Table 4.6 instead of "encode ROM" shown in Table 4.1. This module takes the syndrome bits generated by the module DECODE_STAGE_LAST2 shown in Figure A.16. By using the syndrome bits to look up the decode ROM "Data Bits Error Pattern to Correct" in Figure 2.6 is obtained. The SYND_LAST_CIR in Figure A.17 is the same module as the MAJ_WORD_B_TO_ENC in Figure A.13, except the naming of the input and output signals. Rest of the modules are explained early in this chapter.

In this chapter all modules implementing the MAIN_CIRCUIT were constructed. Because the functionality of this modules are well defined, simulation of these modules will be explained in the following chapter.

IV. SIMULATION RESULTS

In this chapter, first mentor graphics schematic capture tool Design Architect and logic simulation tool QuicksimII are discussed. Simulation inputs and outputs after running the QuicksimII simulations for several modules are given as examples in discussion.

A. MGC DESIGN ARCHITECT

Mentor Graphics tools are used to assist the electronic design Process used in this thesis. Mentor Graphics tools can provide schematic entry, design analysis in IC/PCB layout, test and manufacturing, electronic packaging, and document preparation [Mentor Graphics Idea Station,1989].

A schematic entry tool consists of three main parts. The first part, component libraries, provides files of component and their simulation models used in schematic designs and simulations. NETED (Net Editor-Schematic Capture) creates schematic sheets of components. And SYMED (Symbol Editor) creates component symbols using pins, borders and design properties. Schematic sheets are created by placing components, drawing nets, and adding textual information. It has a variety of different library packages with many library parts to facilitate the design. The routing capability, the gate count, and the delay time in digital simulations provide great insight and analysis capability to the designers.

In this thesis, Xilinx XC4000 library parts associated with Mentor Graphics Design Architect are used to create the design sheets in schematic entries. Besides the basic components, some medium scale integrated circuit components such as counters and RAM bits (16x8 and 16x4 RAM cell-components) provided by Xilinx 4000 family library are also used to build modules. Specifications are explained in Xilinx, Prog. Logic Data Book,1996.

B. SIMULATIONS OF THE DESIGN MODULES

Using the QuicksimII simulations to test and debug the design sheets is straight forward. QuicksimII gives the user the opportunity to go down the design layers to trace and list the individual signal in details. This helps to find out how the signal transitions occur in details in most of the debugging cases. The other benefit of this tool is that it gives the user the capability to save the force values simulation stimulus and the ability to be upgraded later along with the setup and waveform save options.

1. NEWTEST_YEN Simulation

The module NEWTEST_YEN shown in Figure A.2 is the first module in the design to be simulated. The functionality and interfaces of this module are explained in Chapter III. Both the encoding circuit and the registers to keep the serial input for majority vote in the decode cycle are implemented in this module. This module performs the Golay FEC encoding shown in Figure 2.4 using the Equation 2.3. The Generator matrix G shown in Figure 2.3 is stored into the static RAM bits by input signals ENCODE_RAM_IN (11:0), WE, A3, A2, A1, A0. The serial inputs are multiplexed by control signals SERCONT_SIG_0 and SERCONT_SIG_1 into the encoders or into the 49 bit register sets depending on whether it is an encode or decode cycle. The first row of the matrix is all zeros. If the serial input is zero, it is going to encode the first row. There is no effect of serial input zero bit value on the modulo-2 addition. There is a need in implementing the Equation 2.4 to give the code word $\underline{x}$ (1x24 matrix). In this equation, the two message words $\underline{u}$ (two 1x12 bits serial inputs into encoders shown in Figure A.3) look up the rows of both the encode ROM matrices. Each high bit in the message word encodes a corresponding row in the encode ROM. Then these looked up rows are modulo-2 added (or XORed) to yield the two Golay words. They are interleaved with the two message words before being transmitted. The second row of the ENCODE_RAM_IN (input matrix HEX values for encode ROM) shown in Table 4.1 consists of zeros since using the static RAM module introduces additional module delay. The twelve rows of the encode ROM matrix given in Table 4.1 are then entered. The hex values entered into the encode ROM is based on taking the right most bit of each row of the P matrix in G shown in Figure 2.3 as the first bit of hex values given as ENCODE_RAM_IN inputs in the simulations. The P matrix is entered without I matrix, because the 12 bits in the code word will exactly be the same as the message word. A 12 bit shift register (12BIT_SHIFTREG1 in Figure A.3) will keep the serial input message bits. The sixteen rows of the encode matrix are entered by using four address bits (A0, A1, A2, A3). Twelve rows are needed to be addressed in the encode ROM matrices.

Zero bit values are entered into the first and the last two rows of the encode ROM matrix. It is strongly advised to assert a "//globalsetreset" signal for a fraction of a clock period before doing the simulations on the XC4000 family FPGAs. This helps clear the chip modules to eliminate unknown (resistive or high Z) states. The input values of the simulation are summarized in the Table 4.1 as specified above.

CLOCK_PERIOD (nsec.)	**ENCODE_ROM_IN(11:0) (HEX.)**	**A3 (MSB)**	**A2**	**A1**	**A0 (LSB)**
200	000	0	0	0	0
400	000	0	0	0	1
600	C75	0	0	1	0
800	49F	0	0	1	1
1000	D4B	0	1	0	0
1200	6E3	0	1	0	1
1400	9B3	0	1	1	0
1600	B66	0	1	1	1
1800	ECC	1	0	0	0
2000	1ED	1	0	0	
2200	3DA	1	0	1	0
2400	7B4	1	0	1	1
2600	B1D	1	1	0	0
2800	E3A	1	1	0	1
3000	000	1	1	1	0
3200	000	1	1	1	1

Table 4.1. Input values of the encode ROM.

For the write period of the encode ROM (200-3400 ns), WE signal is held high. The control signals SERCONT_SIG_0, and SERCONT_SIG_1, clear signals, transmit enable signals, and shift enable signals shown in Figure A.3. They are given in Table 4.2 when they are at their high states.

CONTROL SIGNAL	HIGH CLOCK PERIOD (nsec.)
SERCONT_SIG_0	4000-6400
SERCONT_SIG_1	6400-8800
CLR_FINXOR_A	100-3800
CLR_FINXOR_B	100-6200
CLR_SERIN	100-150
TR_ENC_OUT	9400-39200
FROM_ENCS	9400-39200
MAJ_VOTE_DONE	100-200
TR_EN1	9800-19600
TR_EN2	19600-29400
TR_EN3	29400-39200

Table 4.2. Values of the control signals at their high clock periods.

The simulation of NEWTEST_YEN module is focused on checking the encoding functionality. The simulation results presented in Figures B.1, B.2 and B.3 are for checking the NEWTEST_YEN module's encoding function.

The encode ROM input signal values are traced between times 200ns and 3400ns as shown in Figure B.1. The WE (write enable) signal is held high as the address bits (A0, A1, A2, A3) are changed to let the encode ROM bits be written.

The values entered into the encode ROM and the first 49bit register bus (REGFIRST_WEN1) transitions' traces are shown in Figure B.2. After the message and the Golay check bits from the two code word bits are interleaved, they are taken into the BUS_A at time 9400ns. This 49 bit sequence is shifted one bit position at each clock cycle into the next REG_NEWSECOND1 module. The interleaved bit sequence is transmitted by TR_SER_ENCOUT output signal shown in Figure B.2. When the FROM_ENCS signal is high, the encoder A and B (ENCODE_IN_WMATs), shift register A and B (12BIT_SHIFTREG1s) values are taken into the BUS_A as shown in Figure B.2.

There are three transmission sequence control signals TR_EN1, TR_EN2, and TR_EN3 as shown in Figure B.3. TR_EN3 signal's being high means that the values in the right REG_NEWSECOND1 module shown in Figure A.3 are transmitted by the TR_SER_ENCOUT signal. TR_EN1 and TR_EN2 signals being low means that the transmission of the bit sequence in REGFIRST_WEN1 and first REG_NEWSECOND1 modules are completed. The BUS_C shows how the bits are shifted at each clock cycle.

SERIAL_IN bit values given in Table 4.3 are being asserted from time 4000ns on to the serial input of the lower ENCODE_IN_WMAT. SERIAL_IN bits are then multiplexed into the serial input of the upper ENCODE_IN_WMAT modules shown in Figure A.3 at time 6400ns and on.

CLOCK @ A (nsec.)	SERIAL_IN (Bit Value)	CLOCK @ B (nsec.)
4000	1	6400
4200	1	6800
4400	0	7000
4600	0	7200
4800	1	7400
5000	0	7600
5200	0	7800
5400	1	8000
5600	1	8200
5800	0	8400
6000	1	8600
6200	1	8800

Table 4.3. Serial Input Values Asserted to Encoder A and B.

The TR_SER_ENCOUT sends the last bit of the code word at time 38800ns since it gets into the 49 bit registers (first into REGFIRST_WEN1) at time 9400ns. As a result the total transmission lasts until 9400+3x49x200 = 38800ns (three times 49 bit shift each at one clock cycle). The NEWTEST_YEN modules behave as expected

2. DECODE_STAGE_LAST2 Simulation

In this simulation, the functionality of the majority voter (shown in Figure A.11), deinterleaving and generation of the syndrome bits in the DECODE_STAGE_LAST2 module shown in Figures A.2 and A.11 are checked.

This module represents the upper portion of the Figure 2.6. The received sequence consists of two twenty four bits code word. Each consists of a message and a check word. The purpose of this simulation is to make sure the majority voting and interleaving are performed correctly. The check bits generated from the received message word can help to find out if received message word has bit errors. If any of the syndrome bits is one, then the message word is in error. Message word is corrected in a way shown in Figure 2.6. The syndrome bits are used to look up the decode ROM to yield the data bits error corrected pattern values. This pattern bits are XORed with the received message word. This will correct the received message word. The encode ROM and decode ROM values are entered in the same way as it is shown in Table 4.1. The row values of the matrices are the only differences in the encode and the decode ROMs. The stimulus values of DECODE_STAGE_LAST2 module are given in Table 4.4, where the encode ROM is the same as the one in NEWTEST_YEN simulation.

Three 49 bit input values are asserted on each bus. They are 2/3 majority voted in a way shown in Figure 2.8. After majority vote, the voted code word bits are interleaved as shown in Figure 2.7. The decode procedure's first part consists of generating syndrome bits as explained above. The three bus values here are picked up in a random manner. Majority clear (MAJ_CLR in Figure A.11) is asserted high before the vote to clear the voter circuit. Majority vote enable signal is asserted when all buses are ready for the vote. When the vote is done, SHIFTAFMAJ_EN signal is asserted to get the voted code words into the received Golay word registers (GOLAY_MAJ_WORD_A. and GOLAY_MAJ_WORD_B) and received message word registers. The received message word is shifted into the encoder module (ENCODE_IN WMAT) by asserting the SHIFT_INTO_ENCS signal for 12 clock cycles. At the end of 12 clock cycles, the intermediate check bits will be ready to be XORed with received Golay check bits. Syndrome bits will be obtained by asserting the GEN_SYNDROME control signal as an enable for XORing mentioned above. The syndrome bits are used to find the data bits error pattern for correction. These pattern bits are XORed with the received message word to yield the corrected message word. The REG_CLR_ASYND signal is asserted high after the correction.

SIGNAL NAME	SIGNAL VALUE	ASSERTED PERIOD (nsec.)
MAJ_CLR	1	100-3400
MAJ_VOTE_EN	1	3800-4000
BUS_A (1:49)	1EBC98234570A	3800-4000
BUS_B (1:49)	1A234598BCDEA	3800-4000
BUS_C (1:49)	189DCA3EF5870	3800-4000
REGCLR_ASYND_XOR	1	100-200
ENCODING_DONE	1	100-200
SHIFT_INTO_ENCS	1	4400-6800
SHIFTAFMAJ_EN	1	4000-4200
CLR_SYNDROME	1	100-200
GEN_SYNDROME	1	7400-7600

Table 4.4. Simulation Input Values and Times When Different From Low Value.

The input values given to these three buses, and to the majority enable signal (MAJ_VOTE_EN) are shown in Figure B.4.

As shown in Figure B.5, when the encode ROM is being written, all other signals stay low. The ROM values in this figure are the same as those encode ROM values used in the NEWTEST_YEN simulation.

The output values are valid at time 7600ns as shown in Figure B.6. The 24 syndrome bits generated are valid after asserting the GEN_SYNDROME signal high between 7400 and 7600ns.

All bus values at time 7600ns are valid after the assertion of SHIFTAFMAJ_EN (shift enable after majority vote). The received message word stays on the VOTED_WORD bus until it is corrected by the data bit error correcting pattern bits. Bus values shown in Table 4.4 begin with ones. These ones are the first bits in these buses. The rest of the values are handled as HEX values. For example, for the BUS_A value 1EBC98234570A, the first 1 is not handled as HEX 1(which is 0001 in 2's complement), but instead it is handled as a single bit(the first bit in 49 bit bus value). Another important point to be taken into consideration is the synchronization of the signals. When they have to go through the same components at the same time, some kind of buffering should be used to synchronize the signals to avoid invalid or unknown results.

3. RAM_BITS_NEW1 Simulation

The RAM_BITS_NEW1 module shown in Figure A.16 takes the syndrome bits to look up decode ROM matrix to yield data bits error pattern for correction as shown in Figure 2.6. The difference between this module and the ENCODE_IN_WMAT module is that the RAM_BITS_NEW1 has a parallel input and serial output syndrome block. Besides, the RAM bits have the input values of the decode ROM matrix instead of the encode ROM matrix that is used previously in section I. The RAM_BITS_NEW1 stimulus force input values other than decode ROM matrix are given in Table 4.5.

SIGNAL NAME	SIGNAL VALUE	ASSERTED PERIOD (nsec.)
SYND_XIN	1	3600-3800
SHIFT_EN	1	4000-6600
CLR_FINXOR	1	100-4200
COUNT_RESET	1	100-4000

Table 4.5. RAM_BITS_NEW1 Force Values.

The syndrome input values are in a sequence of "1 1 0 1 1 0 1 1 1 1 0 1" as S1, S2, S3,..., S12 respectively. They are asserted in parallel between 3600 and 3800 ns. The decode ROM values used to input into the RAM bits are given in Table 4.6.

CLOCK_PERIOD (nsec.)	ENCODE_ROM_IN(11:0) (HEX.)	A3 (MSB)	A2	A1	A0 (LSB)
200	000	0	0	0	0
400	000	0	0	0	1
600	F92	0	0	1	0
800	7C9	0	0	1	1
1000	C76	0	1	0	0
1200	63B	0	1	0	1
1400	C8F	0	1	1	0
1600	8D5	0	1	1	1
1800	B78	1	0	0	0
2000	5BC	1	0	0	
2200	2DE	1	0	1	0
2400	16F	1	0	1	1
2600	F25	1	1	0	0
2800	AE3	1	1	0	1
3000	000	1	1	1	0
3200	000	1	1	1	1

Table 4.6. Input Values of the Decode ROM.

In Figure B.7, decode ROM input values, and the other signal values shown in Table 4.5 are traced. The input values of the decode matrix is taken from the P^T matrix of the parity check matrix H shown in Figure 2.5. The rest of the signals are handled in the same way as they are handled in NEWTEST_YEN simulation section.

The RAM_BITS_NEW1 module generates the valid results at time 6800ns as shown in Figure B.8. The syndrome bits asserted to look up the decode ROM proves to be the same value (579 in HEX as shown in Figure B.8) as the hand calculated value.

In simulations, some vital experience was gained. At very early simulations the force values for stimulus were entered bit by bit, and again and again at each new simulation session. These took a long setup time in simulations. Later in the simulations the hex values for "force files" are used. These force files were saved everytime when the new force values were entered. These were kept for upgrading after debugging of the current schematic designs.

For the functional part of the simulation, the first thing to be considered is to get the implemented function work properly, and produce what is expected. Timing is not a major concern, except that you still can improve the speed by pipelining each stage to see how fast it can get. One of the biggest problems causing the invalid results was that the signals were being asserted asynchronously. This problem is eliminated by using the buffers at the signal paths. The signals coming earlier than the others will then be asserted at the same time.

The other difficulty faced in the simulations was in asserting the bus inputs. This problem is recovered by inputting the ROM matrices in a way shown in this chapter. They (the hex values in simulations of both encoder and decoder) are obtained by entering the ROM values given in Chapter II. Because of some of the ripper and bus naming conventions, the inputs had to be entered just the opposite way. That is the most significant bit value became the least significant bit value, and vice versa. In debugging the big modules that has a couple of modules inside, the simulation input times had to be reprogrammed for the synchronization purposes.

The main purpose of these three simulations is to debug each module, and find out when to assert the control signals to the modules. This will help the design of the controller

V. XILINX FIELD PROGRAMMABLE GATE ARRAYS

Beginning in 1980s, industry was introduced with a new technology to make ASIC designers more competitive. This shortens long acquired testing time, and fabrication-dependent wait period. Using FPGAs one can implement large digital circuits on a single chip. For Xilinx XC4000EX chips, they have up to 125 000 gates, and for XC4062XL it has up to 72K static RAM bits on a single chip [Xilinx, The Programmable Logic Data Book, 1996]. Along with laser programmable chip technology, FPGA technology is among those which grows at the fastest speed to offer industrial solutions to compete with VLSI technologies. The biggest disadvantage with FPGAs is the overall speed concern since the wiring and switching introduce resistive propagation delays as compared to transistor level design. But it is managed to get up to 66 Mhz overall speed in designs with Xilinx FPGAs. Now the FPGAs have more flip flops, and faster switching capabilities, and special clock distribution lines that can run through the whole chip. Faster speed designs are accomplished by pipelining, and by up to date solution database offered to the customers on the Internet. Field Programmable Gate Arrays provides the benefit of custom CMOS VLSI, while avoiding the initial cost, time delay, and inherent risk of a competitor technology, conventional masked gate array. Programmable logic devices provide the benefits of high integration levels without the risks or expenses of semi-custom and custom IC development. Here are some of the benefits of programmable logic devices versus mask programmed gate arrays:

A. Faster design and verification: FPGAs can be designed and verified quickly while the same process requires several weeks with gate arrays. There are no non-recurring engineering (NRE) costs, no test vectors to generate, and no delay while waiting for prototypes to be manufactured. [Xilinx, Programmable Data book, 1996]

B. Design changes without penalty: Because the devices are software configured and user-programmed, modifications are much less risky and can be made anytime, in a matter of minutes or hours, as opposed to weeks it would take with gate arrays. This results in significant cost savings in design and production.

C. Shortest time-to-market: When designing with programmable logic, time to market is measured in days or a few weeks, not months often required when using gate arrays. A study by a marketing research firm concluded that a six month delay in getting to market can cost a product one-third of its lifetime potential profit. With masked programmed gate arrays, design iterations can easily add that much additional time, and more to a product schedule [Xilinx, Programmable Logic Data Book,1996]. FPGAs replaces most discrete and SSI devices, resulting in an 80% reduction in the number of components and a 75% reduction in test time since it does not require test vectors. [Xilinx, programmable Logic Breakthrough '95,1995].

The result of experience gained with two successful FPGA family (XC2000 ,and XC3000) helped the XC4000, XC5000, and XC6000 families to provide a regular, flexible, programmable architecture of Configurable Logic Blocks (CLBs) [Xilinx, Technical Data, XC4000 LCA family, 1990]. They are interconnected by a powerful hierarchy of versatile and abundant routing recourses, and surrounded by a perimeter of programmable I/O blocks on the chip.

The devices are customized by loading configuration data into the internal memory cells. The FPGAs can either actively read its configuration data out of external serial or byte-parallel PROM (master modes), or the configuration data can be written into the FPGA from a host (slave and peripheral modes).

The Xilinx family is supported by powerful and sophisticated software, covering every aspect of the design: from schematic entry to simulation, to automatic block placement and routing of the interconnects, and finally the creation of the configuration bit stream. Since the FPGAs can be re-programmed for unlimited number of times, they can be used in innovative designs where hardware configuration is changed dynamically, or where hardware must be adopted to different user applications. FPGAs are ideal for shortening the design and development cycle, but they also offer a cost effective solution for productive rates well beyond 1000 systems per month. [Xilinx, Tech. Data, XC4000 LCA family

A. XILINX LOGICAL CELL ARRAYS

Xilinx high density user programmable gate arrays comprise three major configurable elements: configurable logic blocks (CLBs), input/output blocks (IOBs), and interconnections. The CLBs provide the functional elements for constructing the user's logic. The IOBs provide the interface between the package pins, and internal signal lines. The programmable interconnect resources provide routing paths to connect inputs and the outputs of the CLBs and IOBs onto the appropriate networks [Xilinx, Technical Data, XC4000 LCA Family, 1990].

Customized configuration is established by programming internal static memory cells that determine the logic functions and interconnections implemented in the LCA device. XC4000 family achieves high speed through advanced semiconductor technology (0.35 micron CMOS fabrication technology), chip density reaches up to 125 K logic gates with system performance of up to 66 MHz speed. It is also 100% Peripheral Component Interconnect (PCI) compliant supporting standard ASIC design flow. Three layers of metal is used in latest XC4000EXs, and extra feature of Select-RAM memory in XILINX chips dramatically improves the system performance, ease-of-use, and overall gate count. This feature allows easy distribution of high performance customized single or dual port RAM functions. [Xilinx, Pro. Log. Data Book, 4-8, 1996]

B. XILINX FAMILY ARCHITECTURE

1. Configurable Logic Blocks

A number of architectural improvements contribute to the Xilinx Family's increased logic density and performance levels. The most important one is a more powerful and flexible configurable logic block (CLB) surrounded by a versatile set of routing resources, resulting in more effective gates per CLB. Each CLB packs a pair of flip-flops, and two independent 4-input function generators. The two function generators offer designers plenty of flexibility, because most of the combinatorial logic functions need less than four inputs. Consequently, the design software tools can deal with each function generator independently, improving the cell usage [Xilinx, XC4000 LCA family, 1990].

Thirteen CLB inputs, and four CLB outputs provide access to function generators and flip-flops. Four inputs are available to each of the two function generators (F1-F4 and G1-G4). These function generators, whose outputs are labeled F' and G', are each capable of implementing any arbitrarily defined Boolean function of their four inputs. The function generators are implemented as memory look up tables, therefore the propagation delay is independent of the function being implemented. A third function generator, labeled H', can implement any Boolean function of its three inputs; F' ,G' and a third input from outside the block (H1). Signals from the function generators can exit CLB on two outputs; F' or H' can be connected to the F output, and G' or H' can be connected to the G output, Thus, a CLB can be used to implement any two independent functions up to four variables or one single function of five variables or any function of four variables together with some functions of five variables, or it can implement even some functions of up to nine variables (Figure 5.1). Implementing wide functions in a single block reduces both the number of required blocks and the delay in the signal path, achieving both increased density and speed [Xilinx XC 4000 LCA family technical data book, 1990].

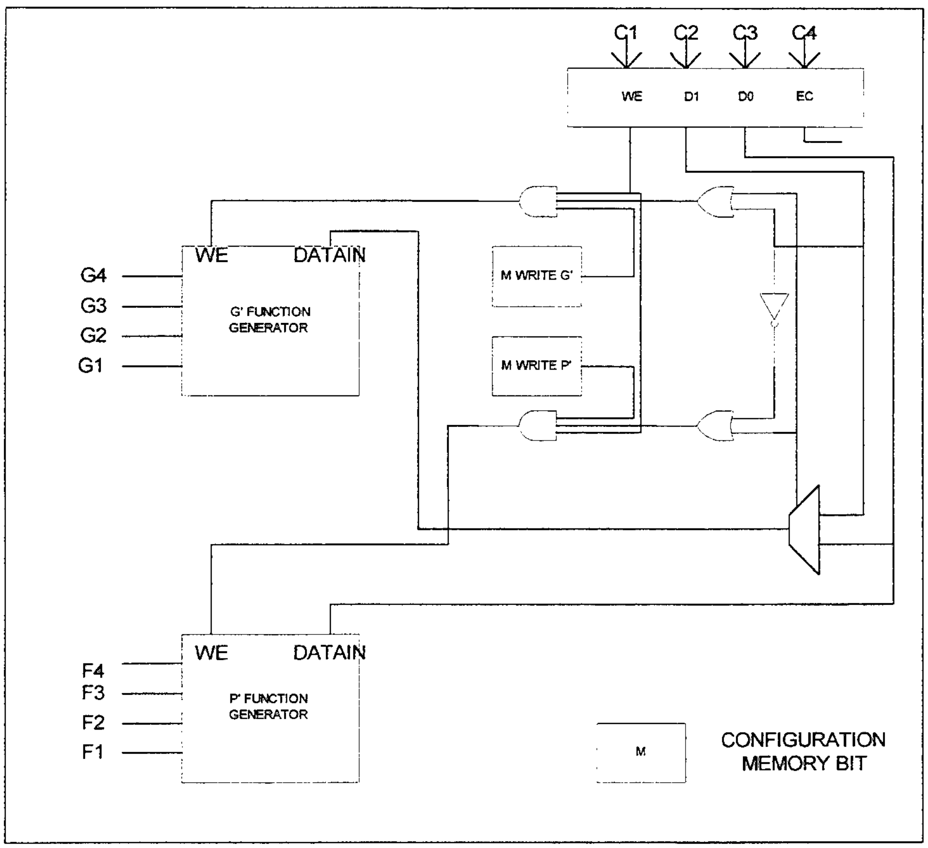

Figure 5.1. CLB Function Generators Has the Ability To Be Used As Read/Write Memory Cells.

The two storage elements in the CLB are edge triggered D-type flip-flops with common clock (K) and clock enable (EC) inputs. A third common input (S/R) can be programmed as either asynchronous set or reset signal independently for each of the two registers; this input can be disabled for either flip-flop. A separate global SET/RESET line (not shown in Figure 5.2) sets or clears each register during power up, reconfiguration, or when a dedicated RESET net is driven active. This RESET input does not compete with other routing recourses; it can be connected to any package pin, providing a global reset input. Each flip-flop can be triggered on either rising or falling edge of the clock. The source of a flip-flop data input is programmable: it is driven either by functions F', G' and H', or Direct In (DIN) block input. The flip-flops drive the Q1 and Q2 of the CLB outputs. In addition, each CLB F' and G' function generator contains dedicated arithmetic logic for the fast generation of the carry and borrow signals, greatly increasing the efficiency and performance of adders, subtracters, accumulators, comparators, and even counters. Multiplexers in the CLB map the four control inputs, labeled C1 through C4 in Figure 5.2, into the four internal control signals (H1, DIN, S/R, AND EC) in any arbitrary manner [Xilinx, XC4000 LCA Family,1994].

Delays in LCA-based designs are layout dependent. While this makes it hard to predict a worst case guaranteed performance, there is a rule designers should consider; the system clock rate should not exceed one third to one half of the specified toggle rate. Critical portions of a design, shift registers and simple counters, can run faster approximately two-thirds of the specified toggle rate [Xilinx, Tech. Data XC4000 LCA family].

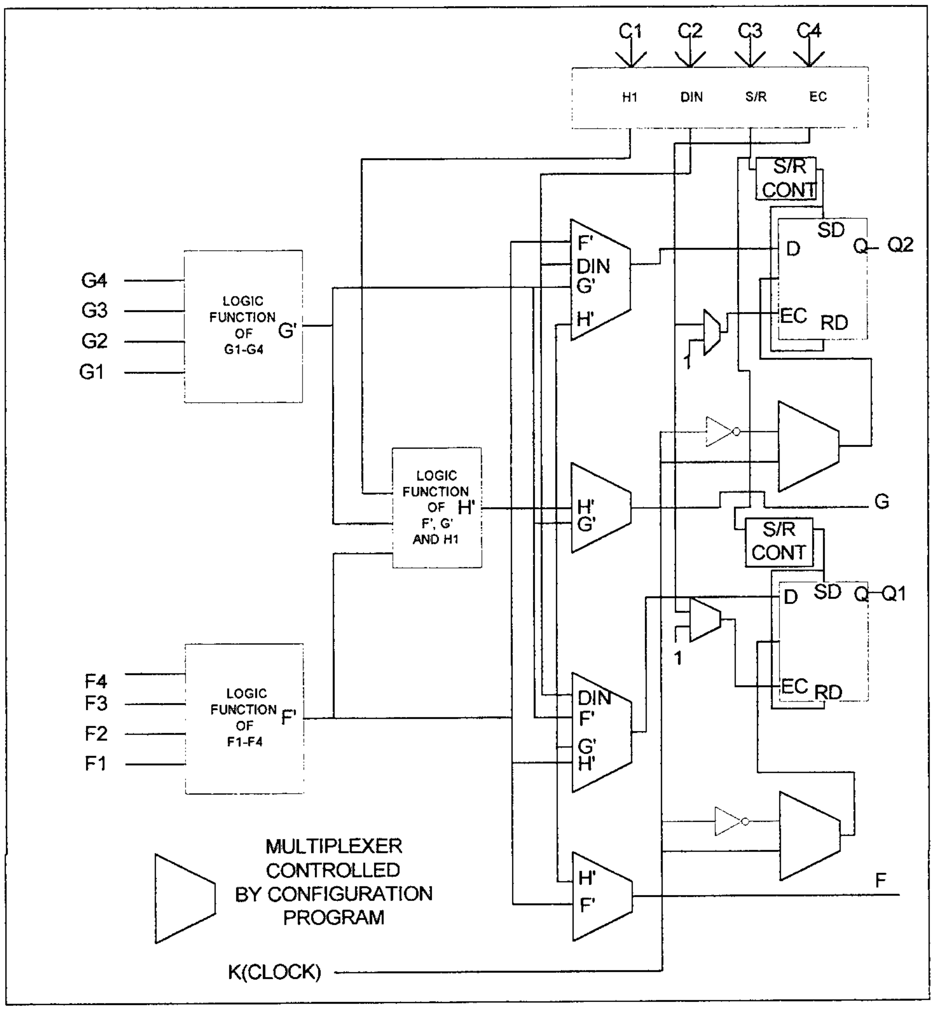

Figure 5.2. Simplified Diagram of XC4000 Configurable Logic Block.

The CLB can pass the combinatorial outputs to the interconnect networks, but can also store the combinatorial result(s) or other incoming data in one or two flip-flops, and connect their outputs to the interconnect network as well. The older programmable gate arrays had to make a choice, either output the combinatorial function or the stored value. From the XC4000 FPGAs on, the flip-flops can be used as registers or shift registers without blocking the function generators from performing a different, perhaps unrelated job. This increases the functional density of the chip [Xilinx, Tech. Data, XC4000 LCA Family, 1990].

When a function generator drives a flip-flop in a CLB, the combinatorial propagation delay overlaps completely with the set-up time of the flip-flop. The set-up time is specified between the function generator inputs and the clock input. This represents a performance advantage over competing technologies where combinatorial delays must be added to the flip-flop set-up time. Each CLB includes high speed carry logic that can be activated by configuration. Each 4-input function generator can be configured as a two bit adder with built in hidden carry that can be expanded to any length. This dedicated adder circuitry is so fast and efficient that conventional speed up methods like carry generate/propagate are meaningless, even at the 16-bit level, and of marginal benefit at the 32-bit level. The fast carry logic opens the door to many new applications involving arithmetic operation, where the previous generations of FPGAs were not fast and/or not efficient enough. High speed address offset calculations in microprocessor or graphic systems, and high speed addition in digital signal processing are two typical applications. As the technology develops, the carry logic helps faster and more efficient counters. In addition, the abundance of flip-flops in the CLBs invites pipelined designs. This is a powerful way of increasing performance by breaking the function into smaller subfunctions, and executing them in parallel, passing on the results through the pipeline. This method should be seriously considered wherever total performance is more important than simple throughput-delay [Xilinx, Tech. Data XC4000, LCA family, 1990].

For years, FPGAs have suffered from the lack of fast and wide decoding circuitry. When the address or the data field is wider than the function generator inputs, FPGAs need level decoding and thus are slower than PALs. In XC3000 there are five bits, in XC4000 family CLBs, there are nine inputs; and any decoder up to 9 inputs is, therefore compact and fast. But there is also a need for larger decoders, especially for address decoding in large microprocessor systems. The XC4000 family has 16 very fast programmable decoders, each with up to 40 inputs. These dedicated decoders are located at chip periphery, four decoders on each chip edge. They accept I/O signals and internal signals as input and generate a decoded output in 10 ns. Each decoder AND gate can also be split into two when a larger number of narrower decoders is required. Very large PALs can be emulated by ORing the decoder outputs in a CLB. This fast decoding feature covers what has been long considered a weakness of FPGAs. Users often resorted to external PALs for simple, but fast decoding functions. The maximum output current specification of today's FPGAs often forces the user to add external buffers, cumbersome especially on bi-directional I/O lines. [Xilinx, Tech. Data XC4000 LCA Family].

2. Abundant Routing Resources

Connections between blocks are made by metal lines with programmable switching points and switching matrices. The globally distributed signal lines have access to any clock or logic input. The designer of synchronous systems can distribute several clocks, and control signals, all over the chip, without having to worry about any clock skew. The horizontal long lines can be used as unidirectional or bi-directional data buses, or they can implement wide multiplexers or wired AND functions. Single-length lines connect the switching matrices that are located at every intersection of a row and a column of CLB. These lines provide the greatest interconnect flexibility, but cause a delay whenever they go through a switching matrix. Double-length lines provide faster signal routing over intermediate distances [Xilinx, Tech Data XC4000 LCA Family,1996].

3. On-Chip Memory

Beginning from XC4000 family LCAs, They have on-chip static memory resources (RAM bits up to 73728 bits for XC4062XL)[Xilinx, The Prog. Logic Data Book, 1996], further increasing the system integration level. And optional mode for each CLB makes the memory look up tables in the F' and G' function generators usable as either 16*2 or 32*1 bit array of read/write memory cells. The F1-F4 and G1-G4 inputs to the function generators act as address lines, selecting a particular memory cell in each look up table. The functionality of the CLB control signals changes in this configuration; H1, DIN, and S/R lines become the two data inputs and the Write Enable (WE) input for 16*2 memory. When the 32*1 configuration is selected, D1 acts as the fifth address bit, and D0 as the data input. The content of the memory cell(s) being addressed are available at the F' and G' function generator outputs, and can exit the CLB through its F and G outputs. Configuring the CLB function generators as read/write memory does not affect the functionality of the other portions of the CLB, with the exception of the redefinition of the control signals. The H' function generator can be used to implement Boolean functions of F' ,G' and D1, and D2 flip-flops can latch the F', G', H', or D0 signals. The RAM bits are very fast: read access time is the same as logic delay (for XC4000 about 5 ns) and write time is 10 ns (again for the same FPGA), both are several times faster than any off-chip solution. Such distributed RAM is a novel concept, creating new possibilities in system design; registered arrays of multiple accumulators, status registers, index registers, DMA counters, distributed shift registers, LIFO stacks, and FIFO buffers. The other important benefit of on board RAM bits is to save gates (such as flip-flops, address line decoders, and etc.) which designer can use to implement more other functions [Xilinx, Tech. Data XC4000 LCA family,1990].

4. Input/Output Blocks

User-configurable I/O blocks (IOBs) provide the interface between external package pins and the internal logic (shown in Figure 5.3). Each IOB controls one package pin and can be defined for input, output , or bi-directional signals. In Figure 5.3, two paths labeled I_1 and I_2 ,bring input signals into the array. Inputs are routed to an input register that can be programmed as either an edge triggered flip-flop or a level sensitive transparent latch. The data input to the register can be delayed to compensate for the delay of the clock. Output signals can pass directly to the pad, or to be stored in an edge triggered flip-flop. OE signal can be used to change output buffer's state to high impedance to implement 3-state buffers or bi-directional I/O. Slew rate control signal is used to minimize power bus transients when switching non-critical signals. The output buffers are capable of sinking 12mA; two adjacent buffers can be wired-ANDed

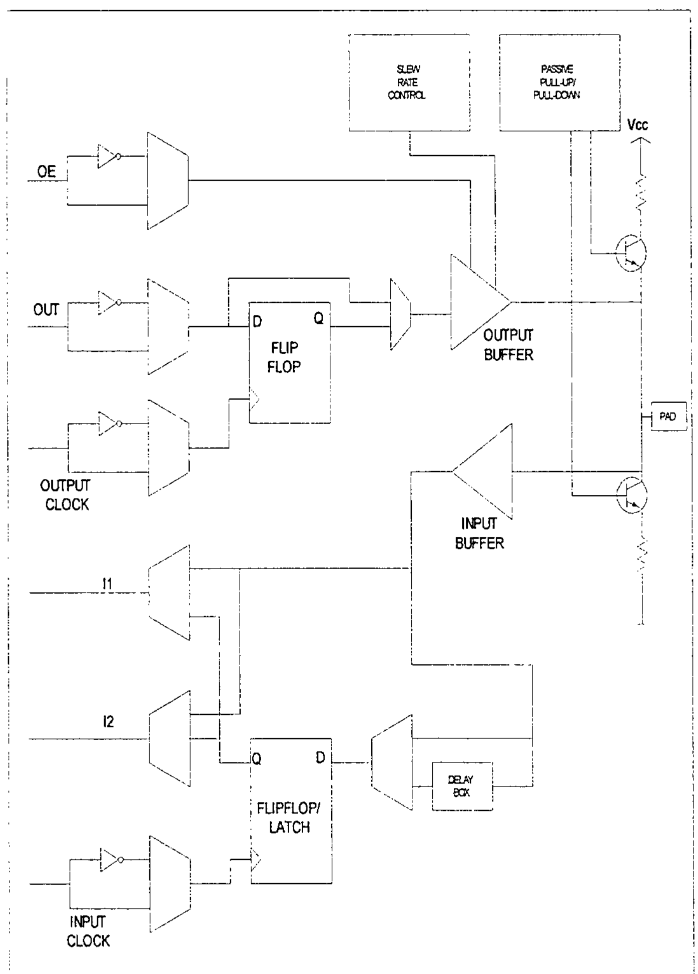

Figure 5.3 Block Diagram of XC 4000 Input/Output Block.

externally to sink up to 24mA in order not to use buffers outside the package, in this case it can support short buses on any card (like PC board). Programmable pull-up and pull-down resistors are useful to connect unused pins to V_{cc} or ground to minimize the power consumption. And it is compatible with IEEE standard 1149.1 for boundary scan testing, permitting easy chip and board level testing of LCA-based applications[Xilinx, Tech. Data XC4000 LCA Family, 1990].

5. Programmable Interconnect

All internal connections are composed of metal segments with programmable switching points to implement the desired routing. An abundance of different routing resources is provided to achieve efficient automated routing. The number of routing channels is scaled to the size of the array, i.e, it increases with array size. CLB inputs and outputs are distributed on four sides of the block (beginning from XC4000 FPGAs), providing routing flexibility. There are three main types of interconnect, distinguished by the relative length of their segments: single-length lines, double-length lines, and long lines. The number of routing channels varies with the array size. The routing scheme was designed for minimum resistance and capacitance of the average routing path, resulting in significant performance improvements. The single-length lines are a grid of horizontal and vertical lines that intersect at a switch matrix between each block. Each switch matrix consists of programmable n-channel pass transistors used to establish connections between the single length lines. And it can be routed one way or multiple ways depending on the branches required. Single length lines are used to conduct signals within a localized area and to provide branching for nets fanout greater than one [Xilinx, Tech. Data XC4000 LCA Family].

Double-length lines provide the most efficient implementation of intermediate length, point to point interconnections. Long lines form a grid of metal interconnect segments that run the entire length or width of the array. Additional long lines in the vertical plane can be driven by special global buffers, designed to distribute clocks and other high fanout control signals throughout the array with minimal skew. Long lines are intended for high fanout time, critical signal nets. Communication between long lines and single-length lines is controlled by programmable interconnect points at the line intersection by six pass transistors to route signal to where it is needed [Xilinx, Tech Data XC4000 LCA family, 1990]. Double length lines do not connect with other lines, and beginning with the XC4000EX quad lines, introduce higher communication or faster propagation capabilities.

6. Taking Advantage of Reconfiguration

All Xilinx families of LCAs can be reconfigured to change logic functions while resident in the system. This gives the system designer a new degree of freedom that is not available with any other type of logic. Hardware configuration can be changed as easily as software. Design updates and modifications are easy. An LCA device can be reconfigured dynamically to perform different functions at different times. Reconfigurable logic can be used to implement system self diagnostics, create system capable of being reconfigured for different environments or operations, implement dual-purpose hardware for a given application. As an added benefit, use of reconfigurable LCAs simplifies hardware design and debugging and shortens a product's time-to-market [Xilinx, Tech. Data XC4000 LCA Family,1996] .

7. The Xilinx Development System

The powerful hardware features of FPGA family require a powerful, easy to use development tool, and Xilinx provides an enhanced tool: Xilinx Automated CAE Tool (XACT). And XACT Design Manager (XDM) simplifies the selection of command line options with pull-down menus and online documents. Design sheet can be entered using schematic capture software or industry standard interfaces such as EDIF. The XACT development system has interfaces with a number of powerful design environments, and it supports more than 100 packages (like Mentor Graphics, ORCAD, FutureNet, VIEWLogic, Cadence, Synopsys, etc.).

C. CONCLUSION

In this thesis study, the XC4003 FPGA environment is used for the simulations of the individual modules. It is concluded that, the XC4003 gate count is not sufficient for the MAIN_CIRCUIT design even before adding the controller. The XC4007 or a higher XC4000 family member should be chosen to satisfy the needs.

As a result FPGAs are used with high performance along with price leadership for high volume applications like Plug-and-play cards, ATM

(Asynchronous Transfer Mode) cards, for wireless communication, CDROMs, SATCOMs, Teleconferencing and DSP applications successfully.

VI. CONCLUSION AND RECOMMENDATIONS

In this thesis study, the golay encoder/decoder is implemented in the Mentor Graphics schematic entry tool Design Architect. The simulations for some individual modules are done in QuicksimII for Xilinx XC4003 FPGA environment. The controller part of the design is to be implemented in a later study according to the simulation results obtained in this study (as explained in Chapter IV). After the design is completed, the migration of the coder into an FPGA chip is to be done. It is necessary to do timing simulations of the coder for the FPGA demo board. The FPGA that will be used to simulate the coder should be a XC4007 or higher gate capacity chip depending on the implementation of the controller.

During the real environment execution on the demo board, the encode and decode ROM inputs need to be done only once at the beginning. The static RAM bits can keep the ROM values entered. Therefore, the input/output pads can be assigned to other I/O signals. The static RAM bits of the XILINX architecture provided faster encoding, and decoding capability, but it took time to understand how to include it in the design process.

For a faster schematic design of the coder, flip flops can be used at several stages by dividing the modules into equal design partitions for pipelining. In the XC4000 family there are a number of options that can offer enough gate counts for designs requiring large number of gates for high speed pipeline operation.

Modular design approach helped a lot in the implementation and the debugging of the modules and layers of the design hierarchy. Moreover it helped the understanding of the function partitions and interfaces.

For future implementation of the controller, the first approach is to define the partitions of states independent of each other. After debugging each state transition diagram of the partitions, they all can be put together to perform the overall state transition diagram. By using this state table and an algorithm (such as one-hot) the controller can be implemented. The second approach is to use a synthesis tool (like VHDL or Verilog in one of the EDA tools) by writing the states of the controller required in the design in high level languages. After converting it into schematic sheet, it can be imported into the FPGA chips.

Before designing the controller and exporting the design into an FPGA chip, it is important to study the State Machine Design Chapter in the Digital Design Book [John F. Wakerly,1994], Xilinx XC4000 FPGA training course manuals, and the XILINX "User Guide and Tutorials" book to save time before facing the potential problems.

The lack of experience with the schematic capture and simulation tools is part of the difficulty in this work. Customer services of the design programming environment and direct support people were often not available. This problem resulted in slow down during the thesis study.

APPENDIX A.

CIRCUIT SCHEMATICS

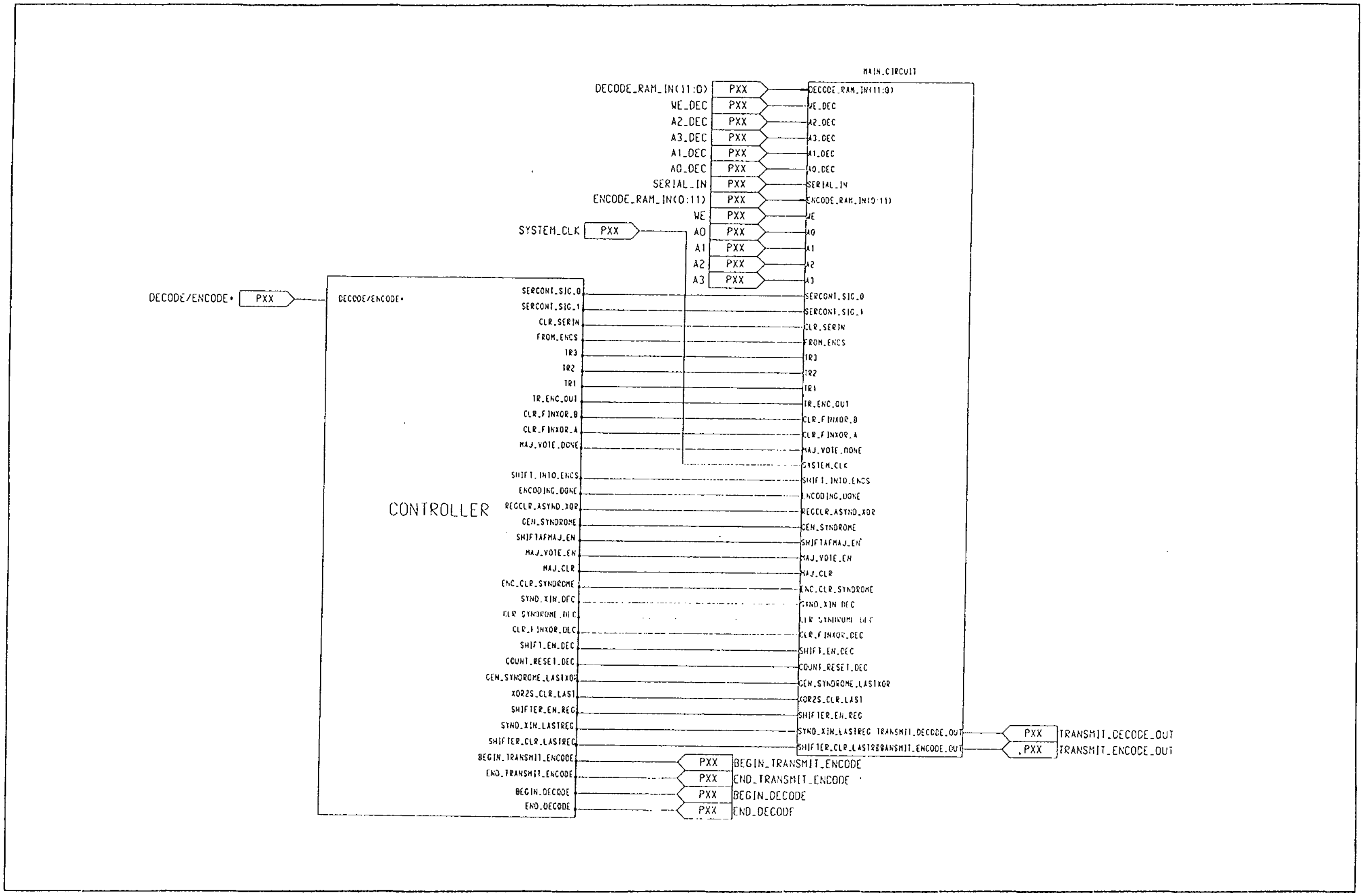

Figure A.1. GENERAL_ENCDEC, General Encoder/Decoder.

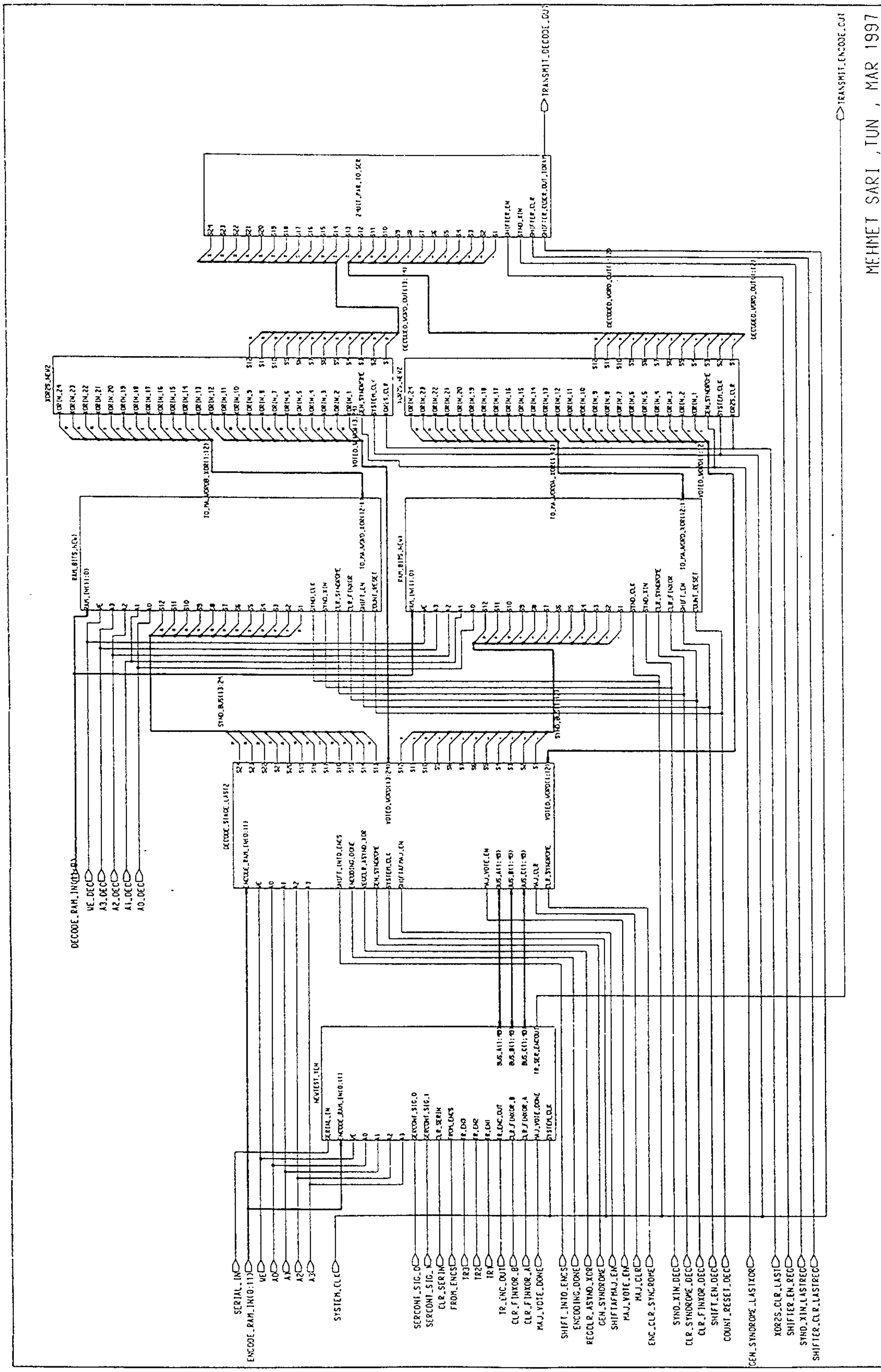

Figure A.2. MAIN_CIRCUIT, Main Circuit.

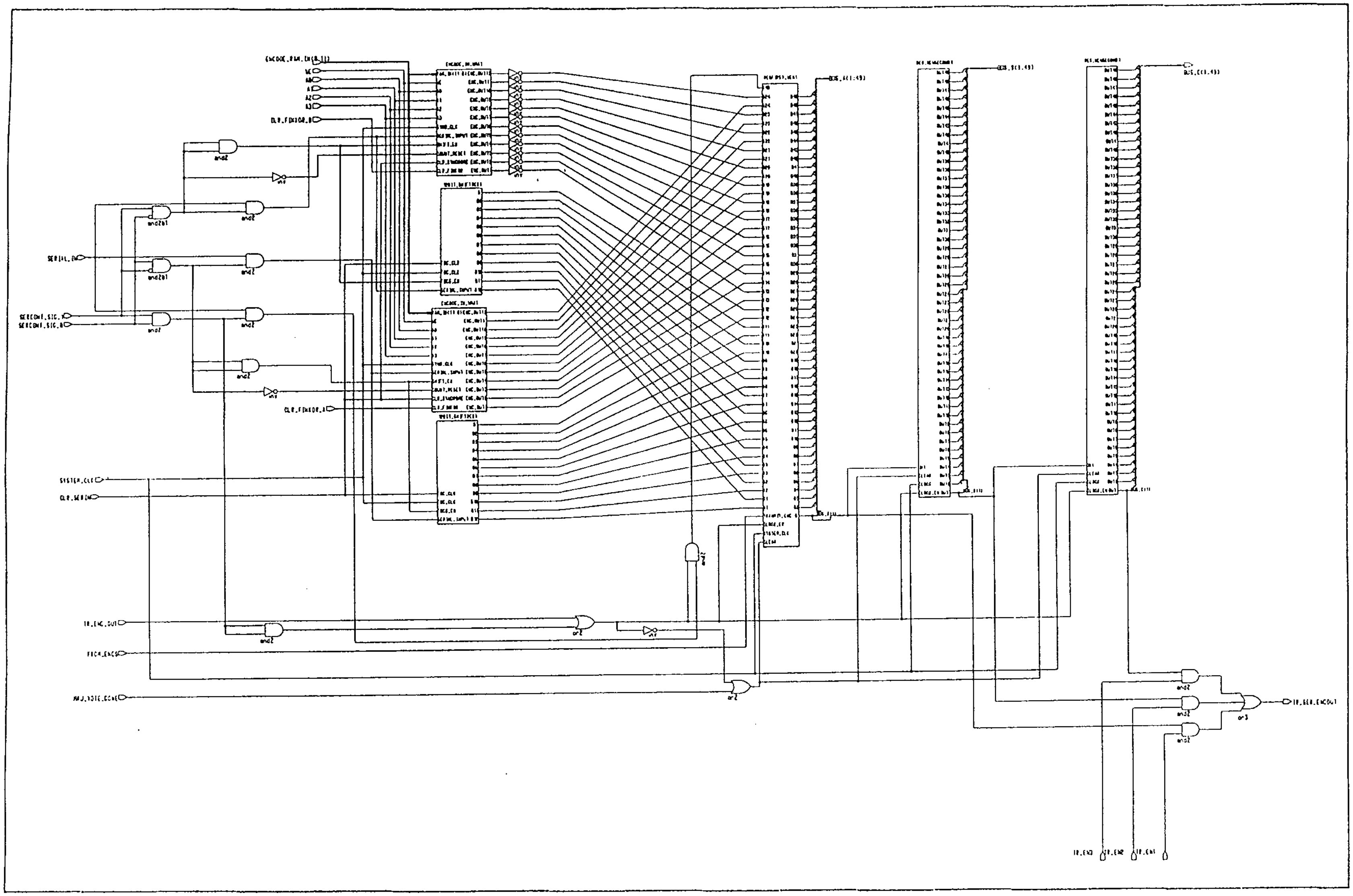

Figure A.3. NEWTEST_YEN, Encoder Module and Register Set for Vote.

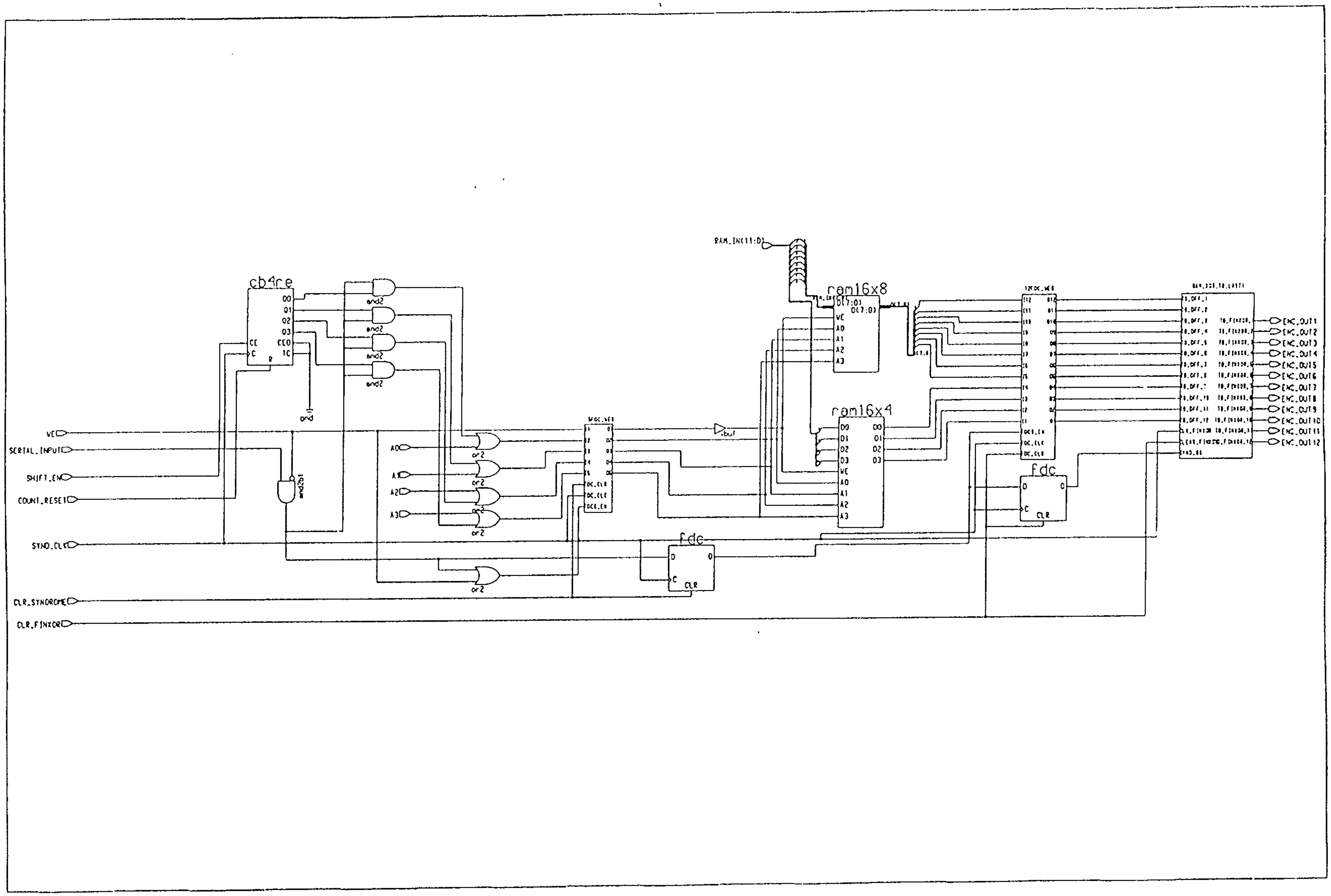

Figure A.4. ENCODE_INWMAT, Encoding Serial Input with an Encode ROM Matrix.

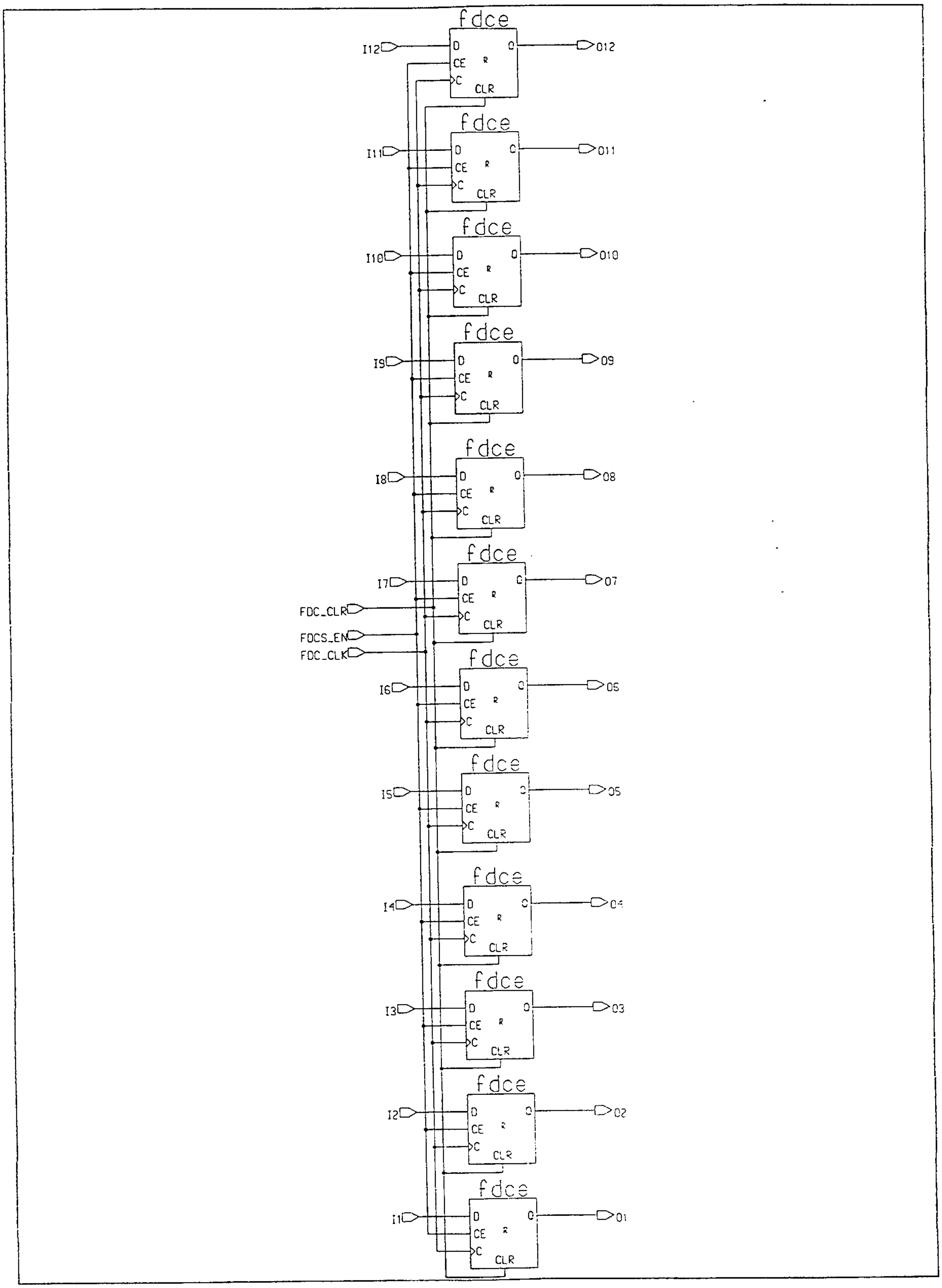

Figure A.5. 12BIT_SHIFTREG1, 12 Bit Shift Register.

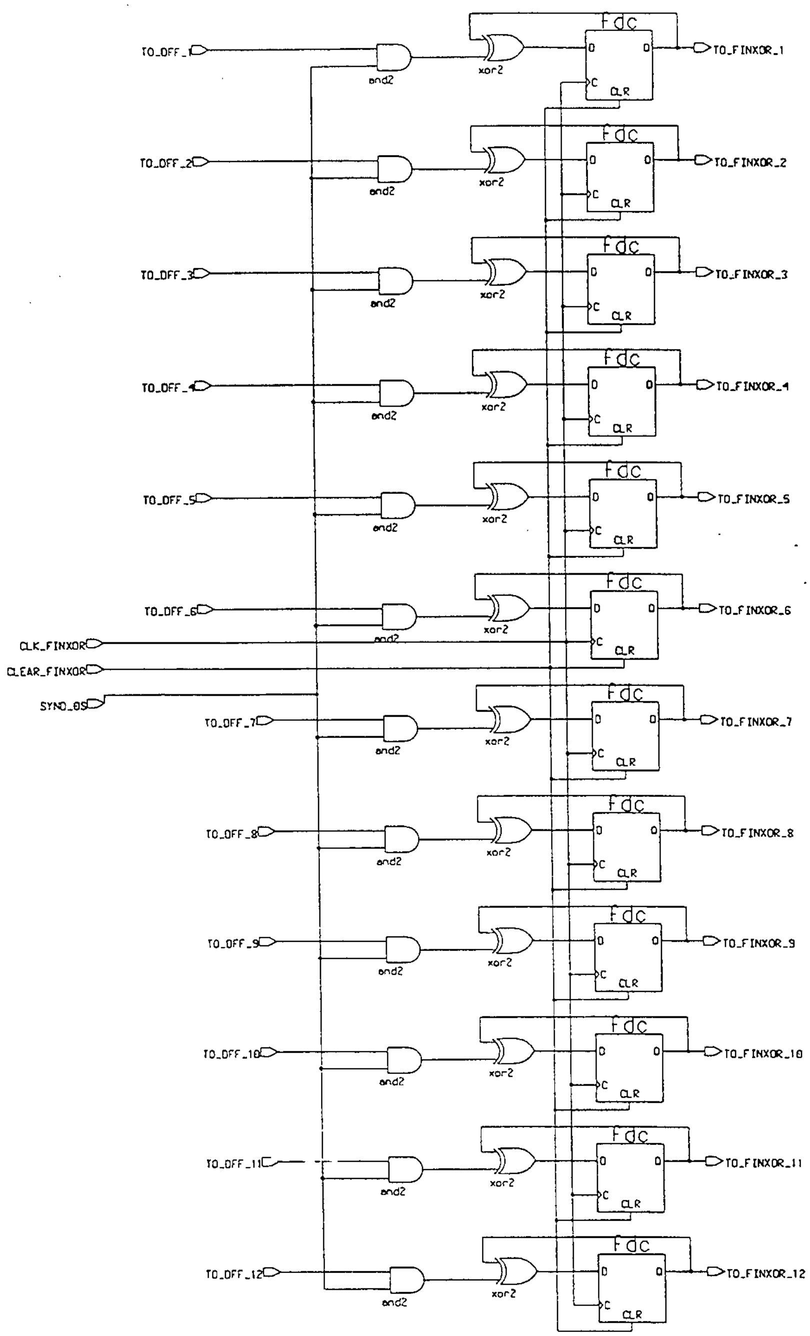

Figure A.6.RAM_XOR_TO_LAST1, Looked up Encode ROM Rows XOR Module

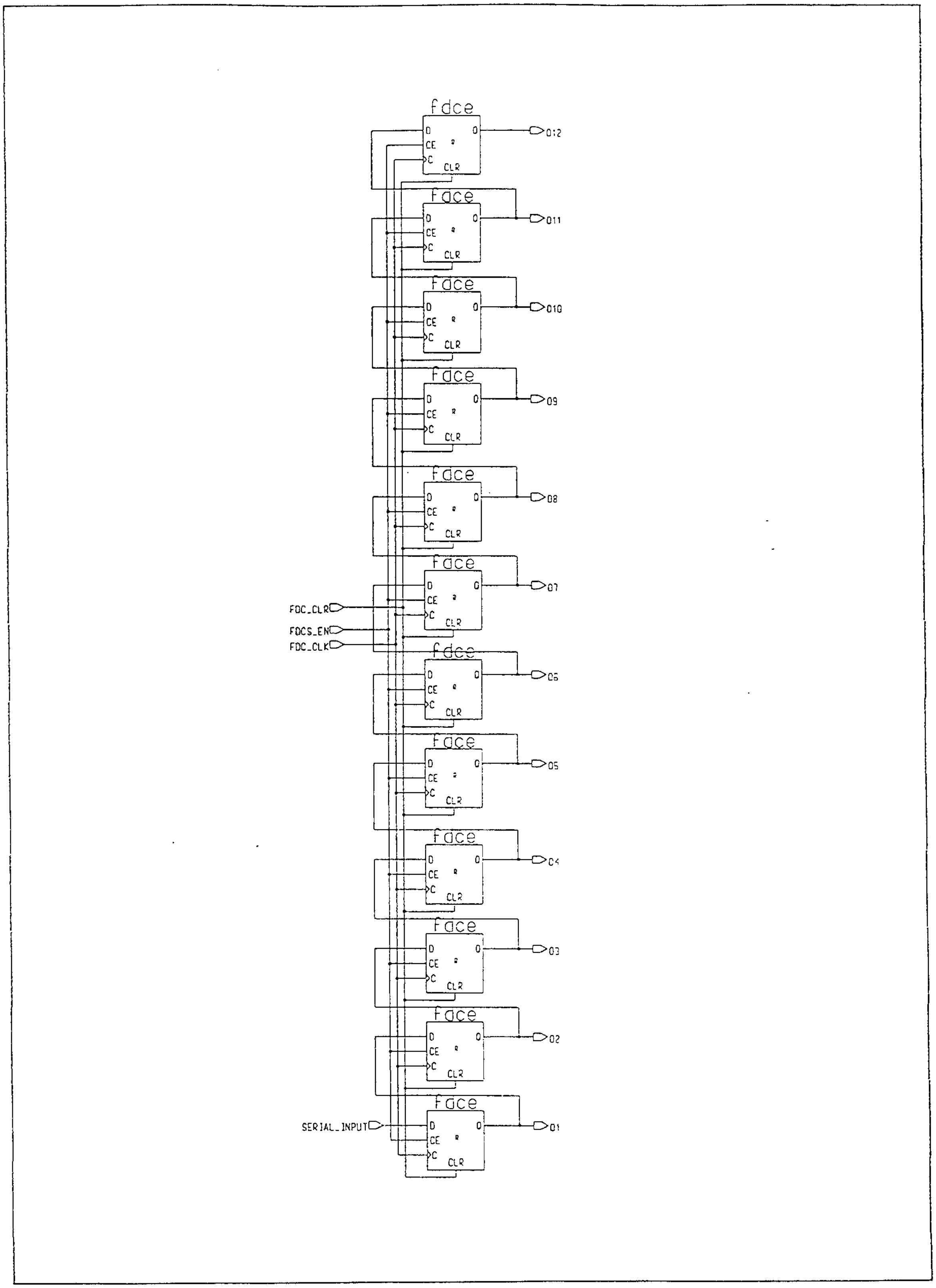

Figure A.7. 12FDC_WES, 12 D-Flip-Flop with Enable.

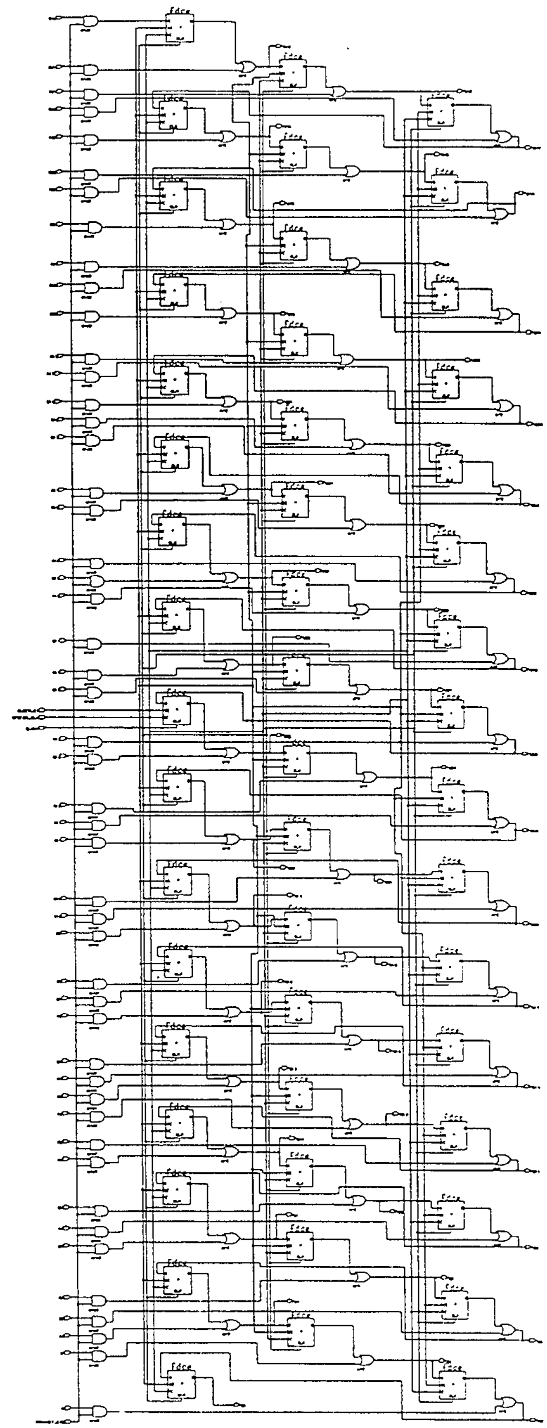

Figure A.8. REGFIRST_WEN1, First Column 49 Bit Register Module.

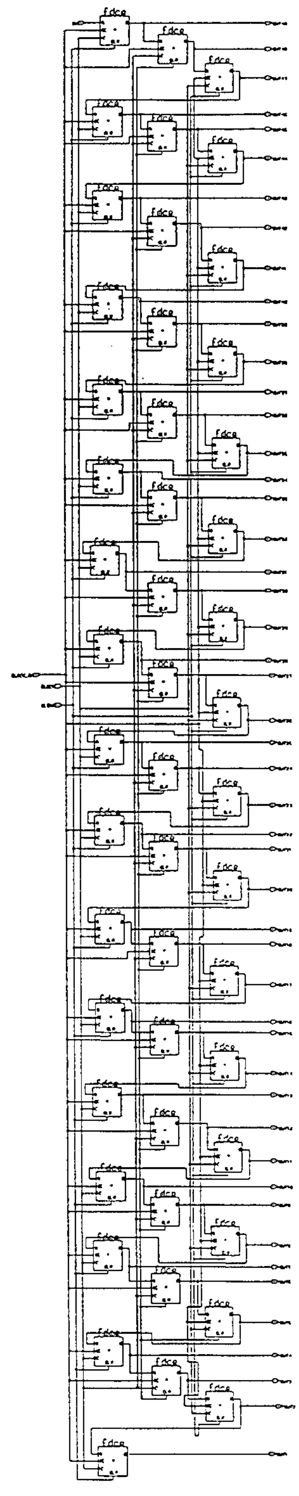

Figure A.9. REG_VEWSECOND1, Second and Third Column Register Module.

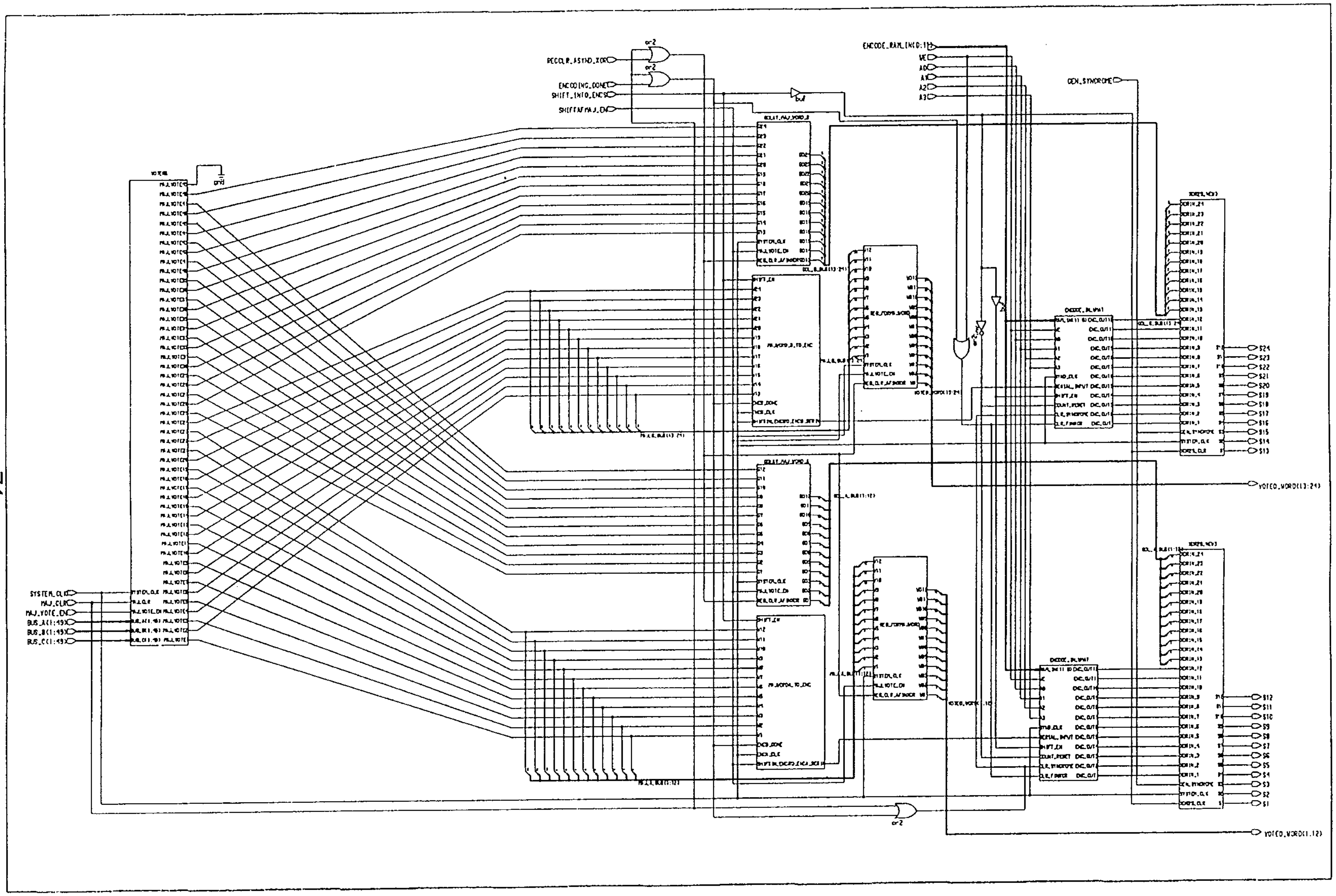

Figure A.10. DECODE_STAGE_LAST2, Majority Voter and Syndrome Generator Module.

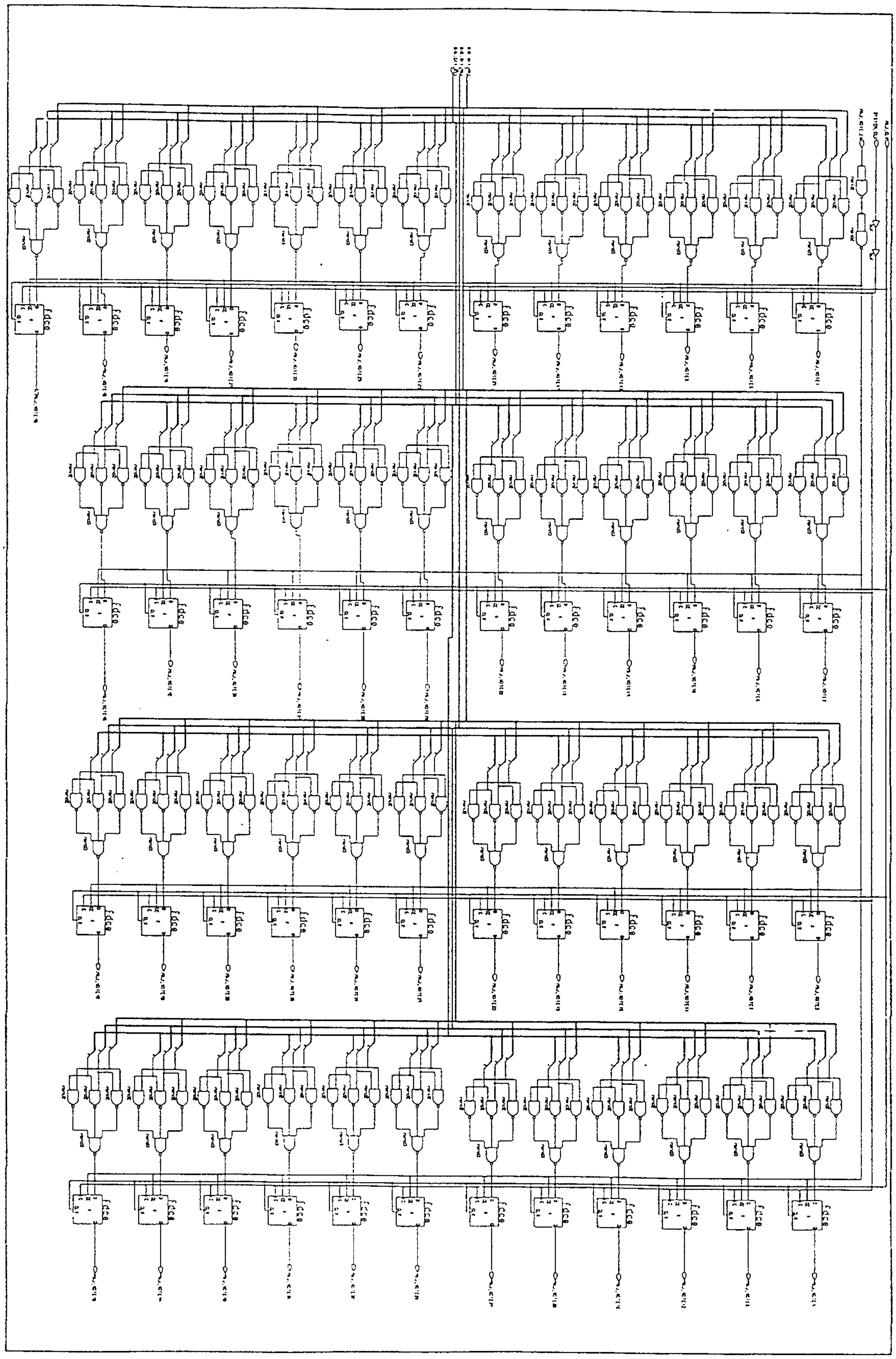

Figure A.11. VOTER3, Majority Voter.

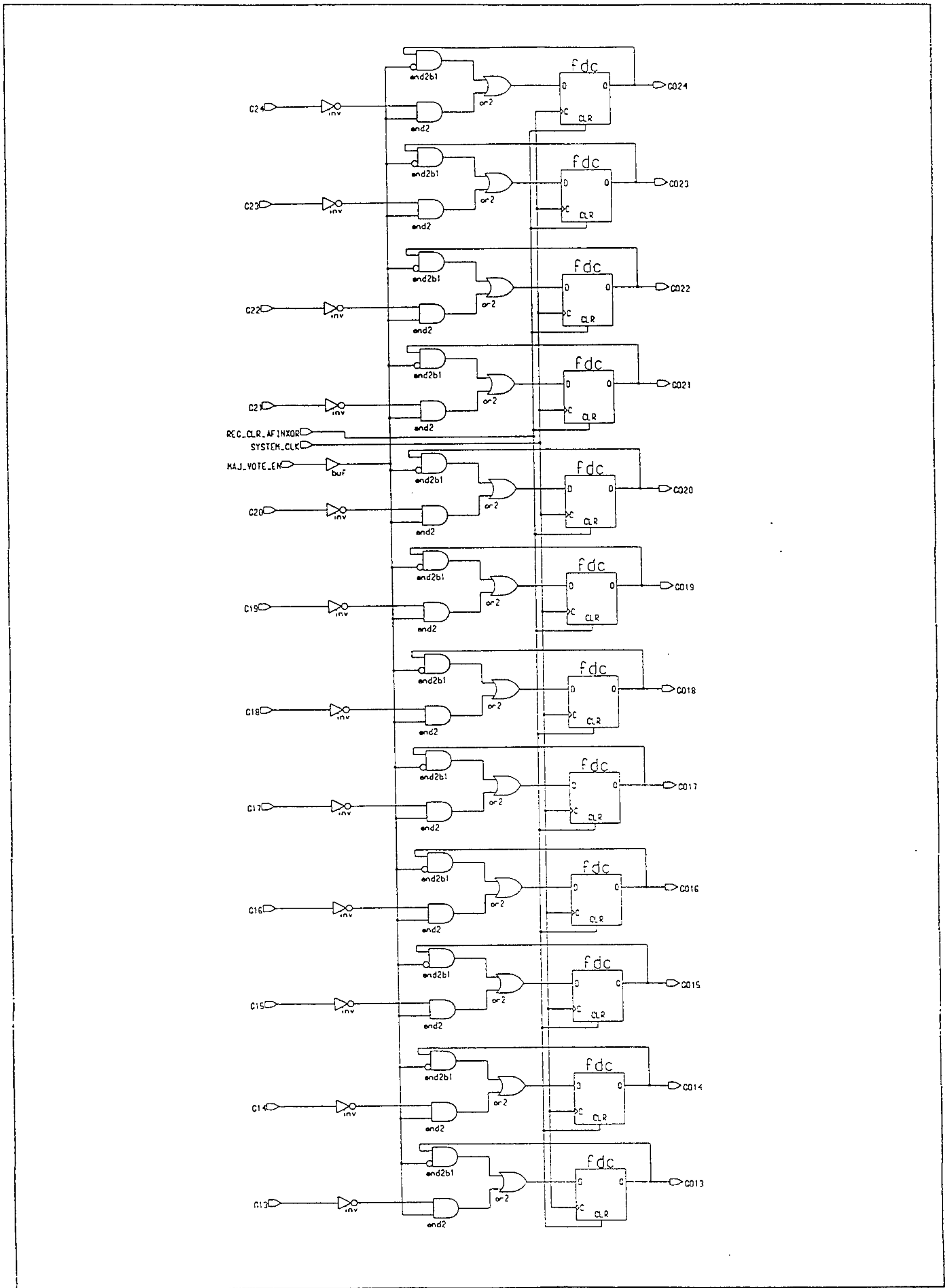

Figure A.12.GOLAY_MAJ_WORD_B, Received Golay Word Register.

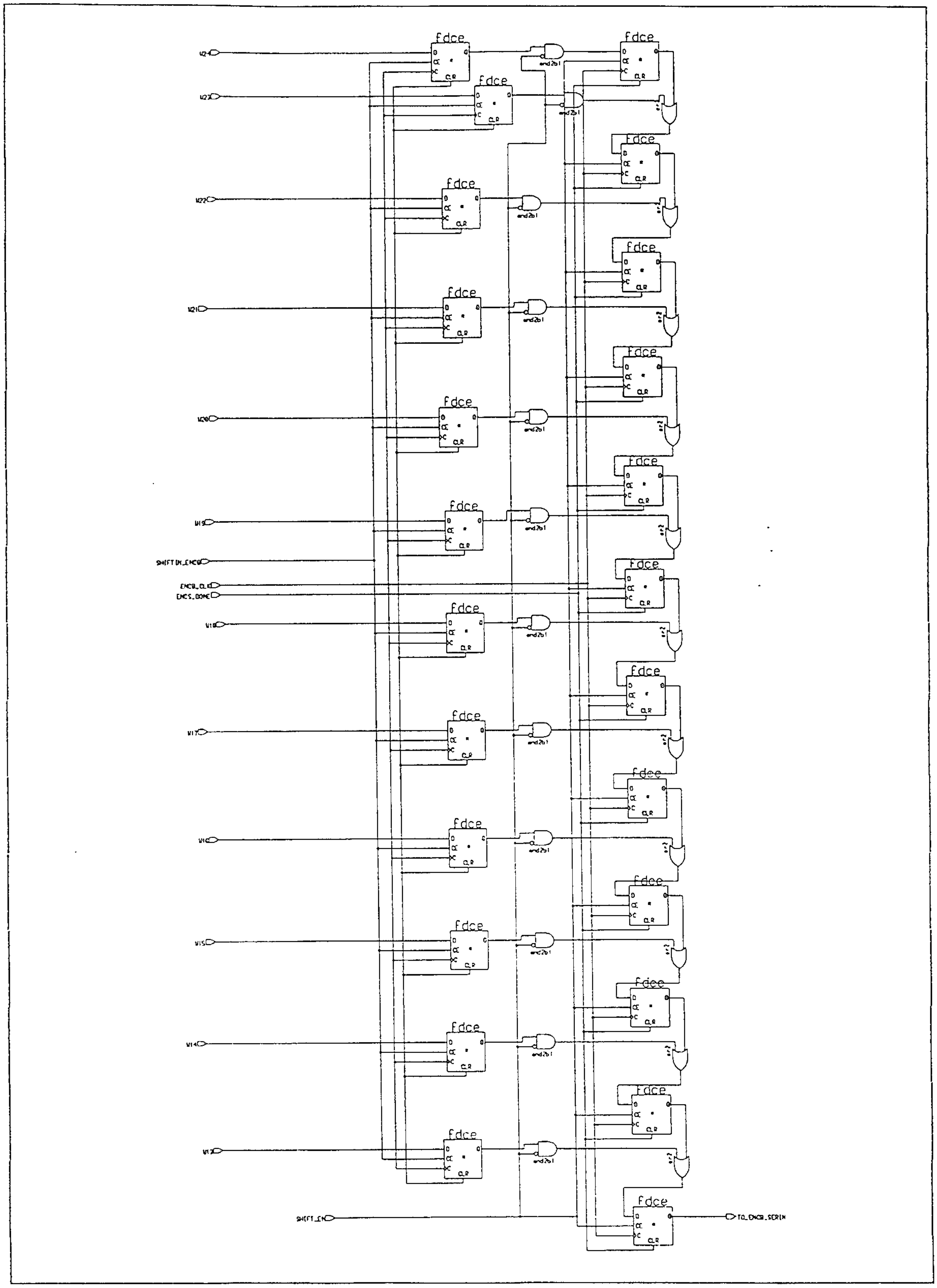

Figure A.13. MAJ_WORD_B_TO_ENC, Received Message Word Register.

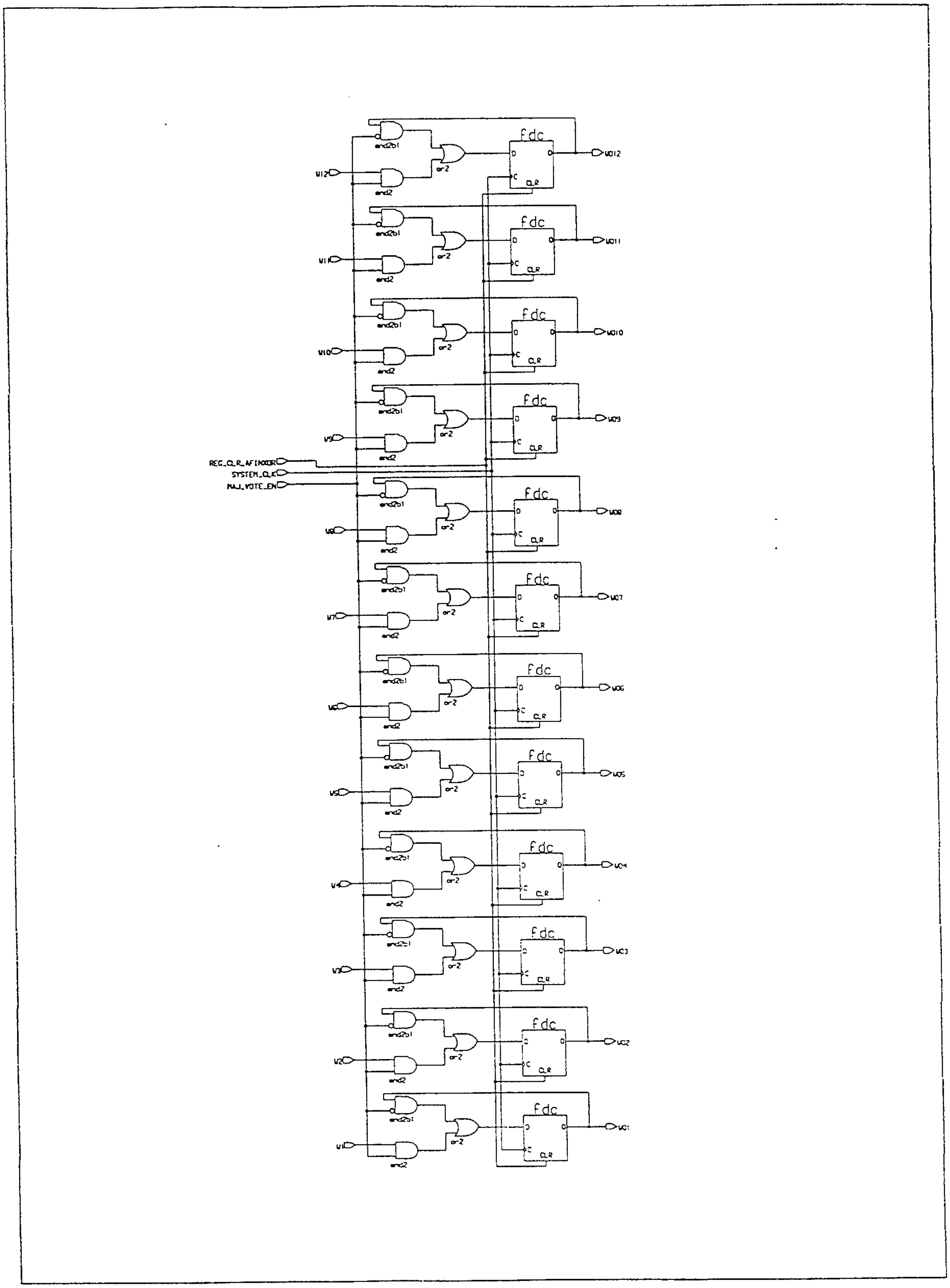

Figure A.14. REG_FORMAJ_WORD, Register for Majority Word.

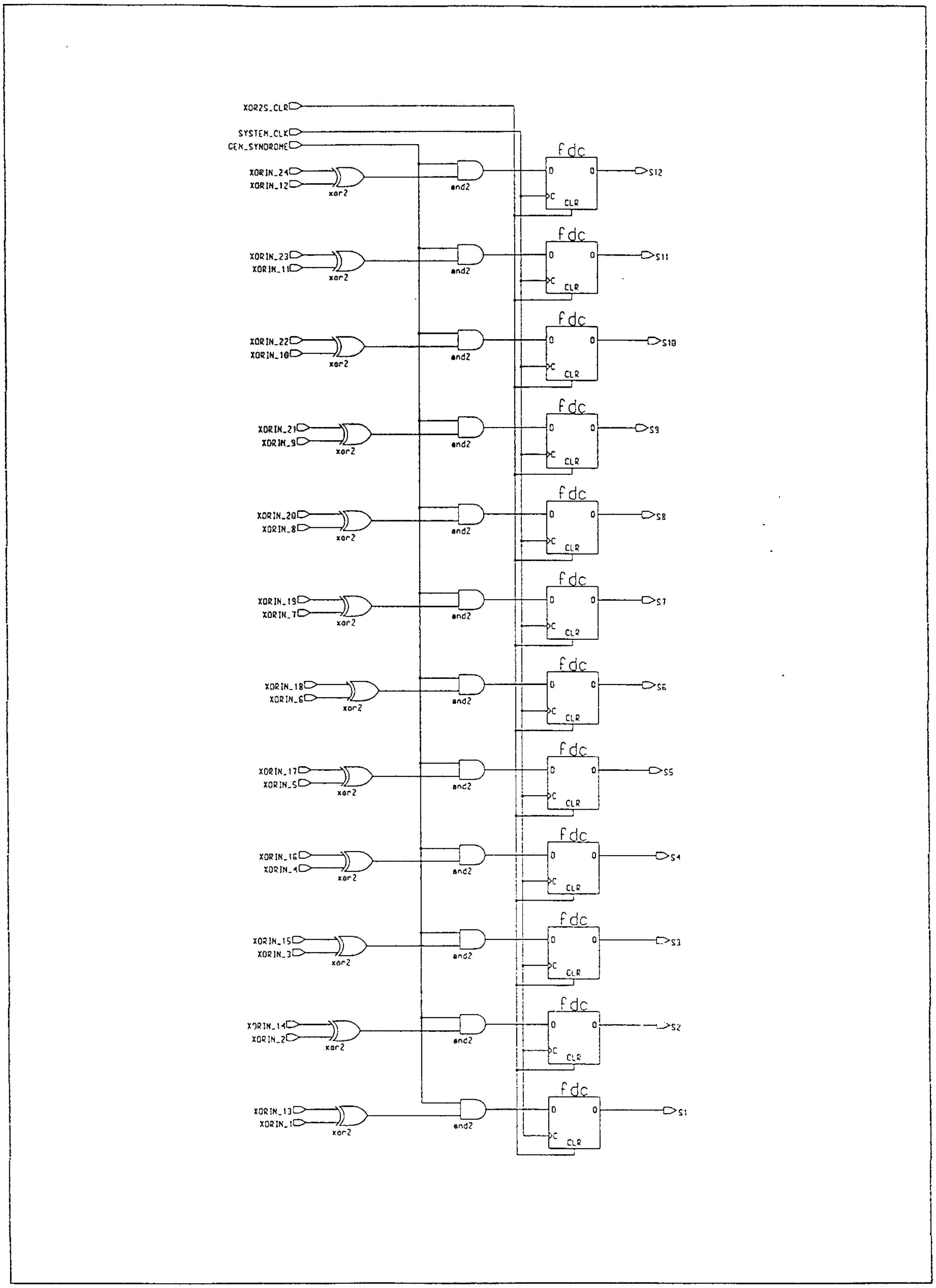

Figure A.15. XOR2S_NEW3, Syndrome Bit Generator XOR Module.

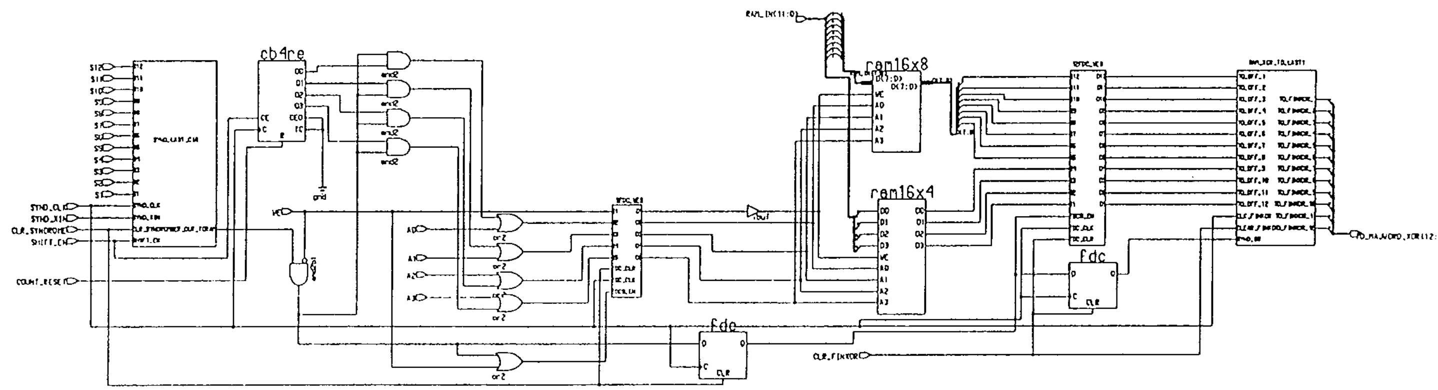

Figure A.16. RAM_BITS_NEW1, Decode ROM Look up Module.

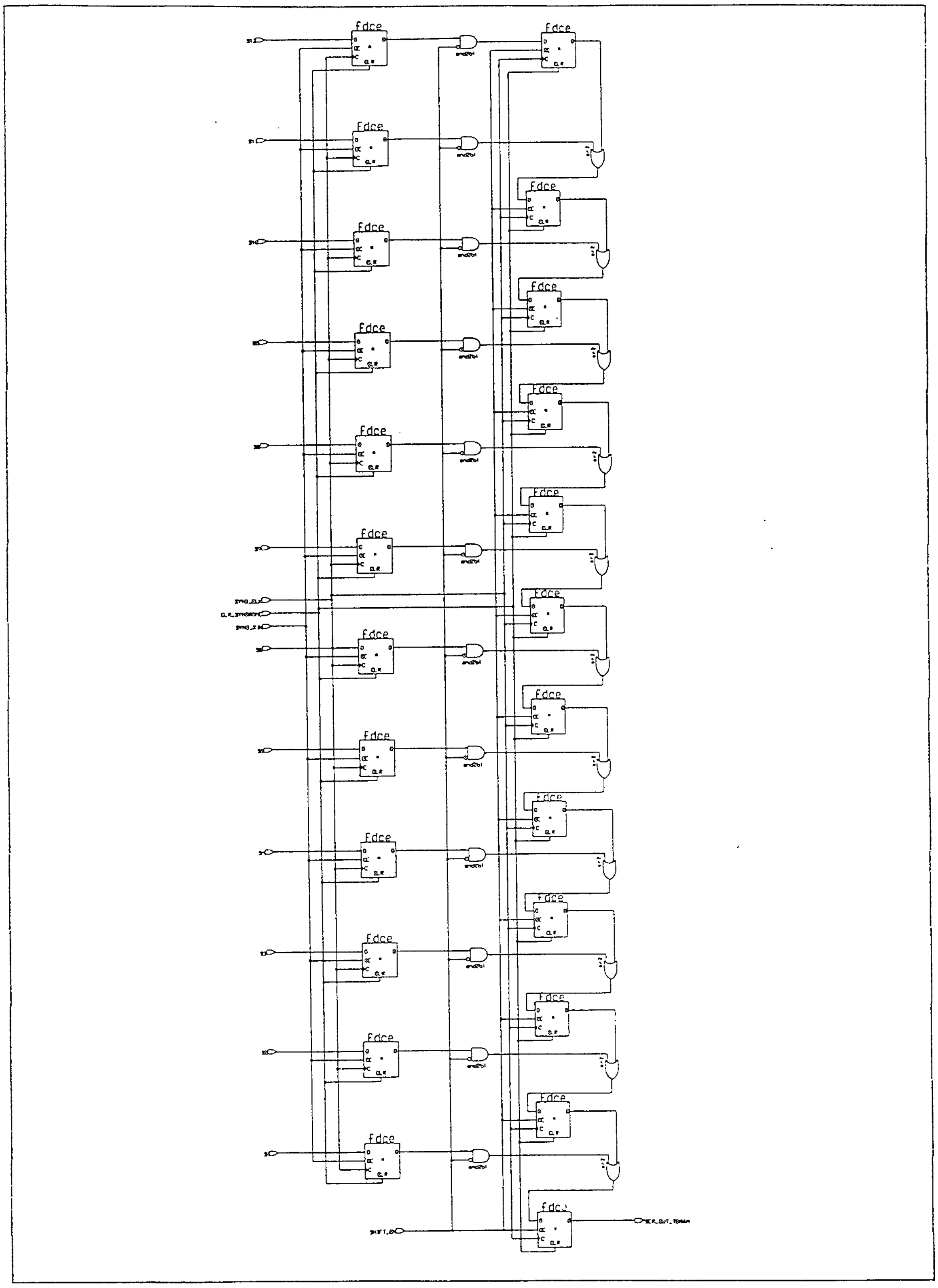

Figure A.17. SYND_LAST_CIR, Syndrome Bits Shift Module.

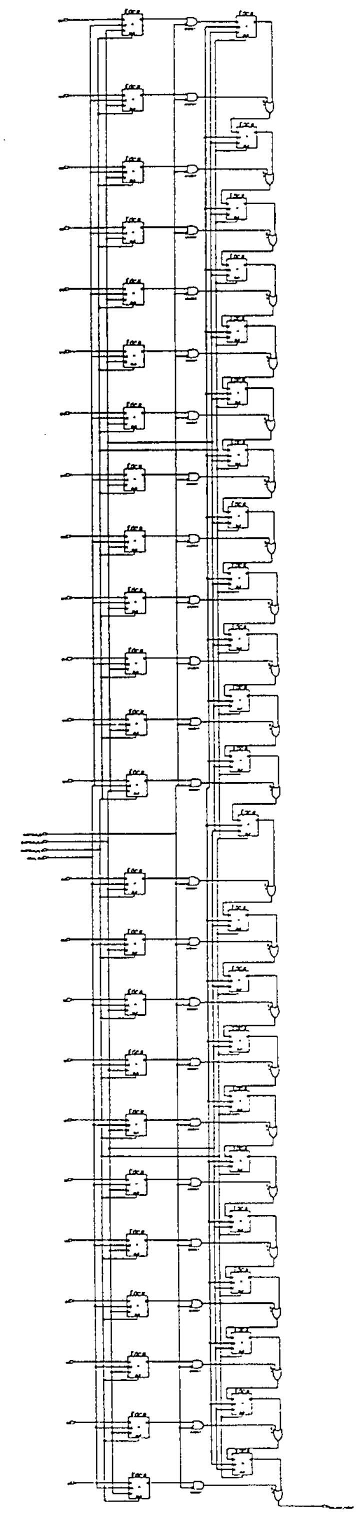

Figure A.18. 24BIT_PAR_TO_SER, 24 Bit Massage Words Shift Module.

APPENDIX B.

SIMULATION SCHEMATICS

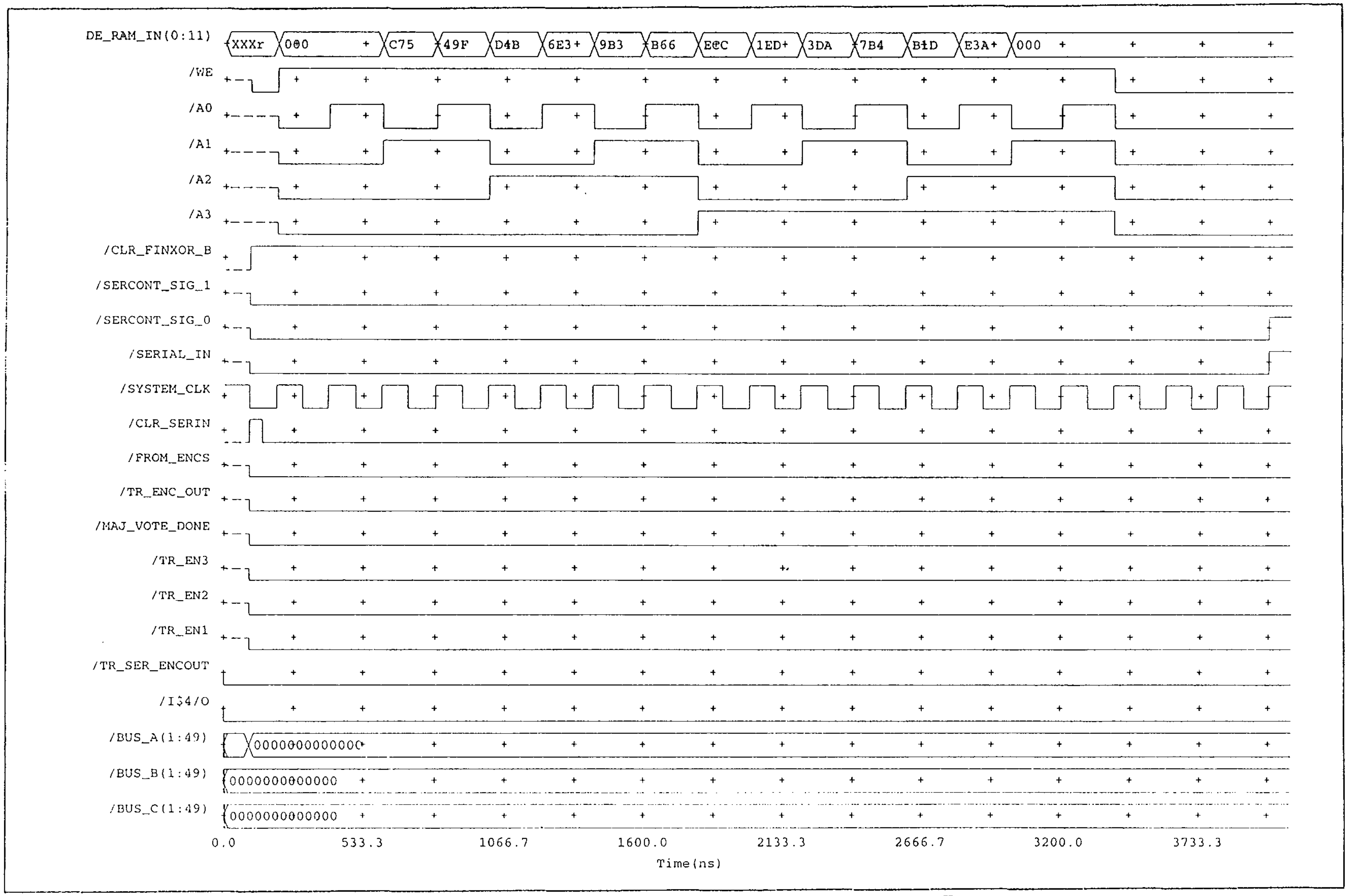

Figure B.1. NEWTEST_YEN Simulation Inputs Trace.

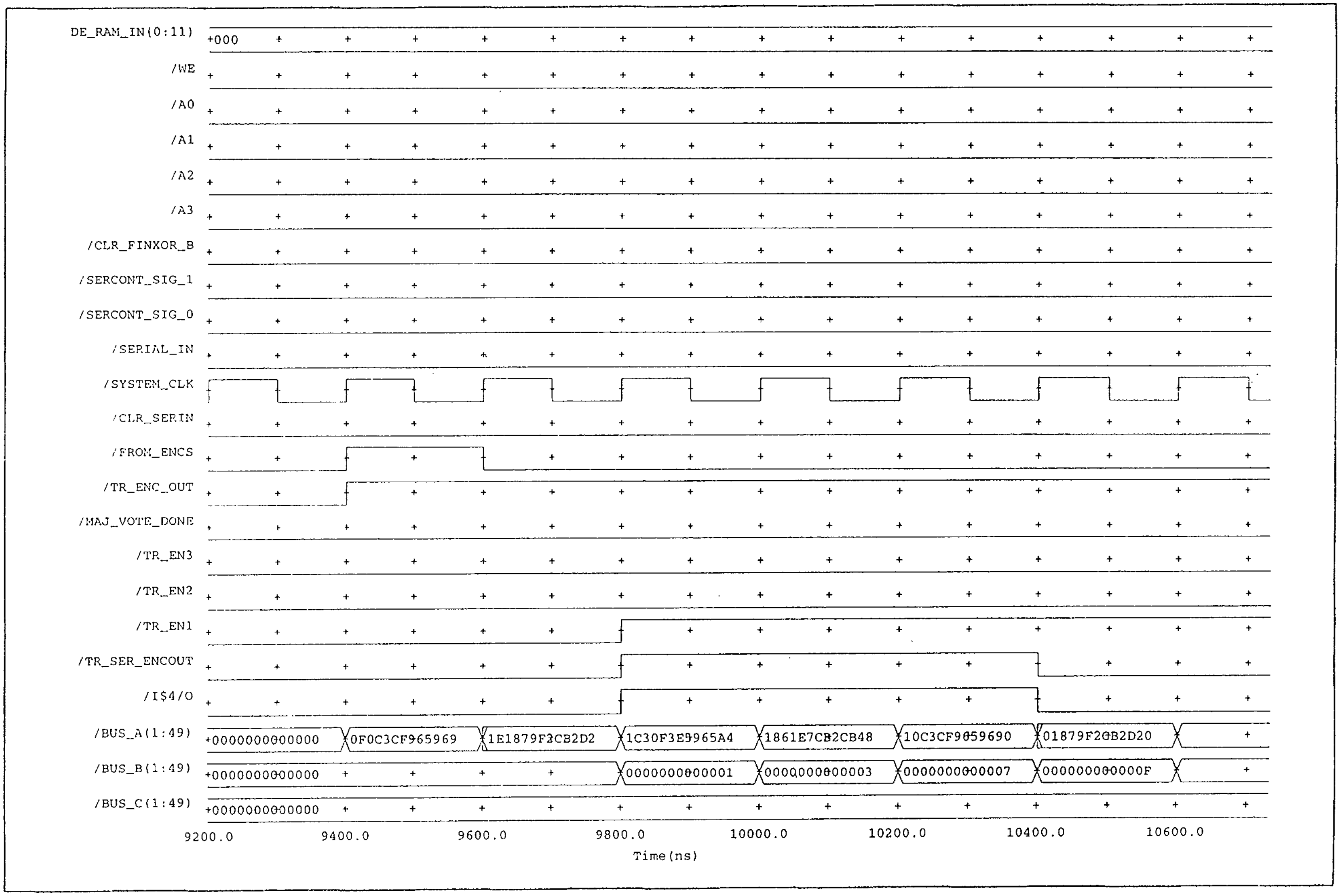

Figure B.2. NEWTEST_YEN Registers Shift Trace.

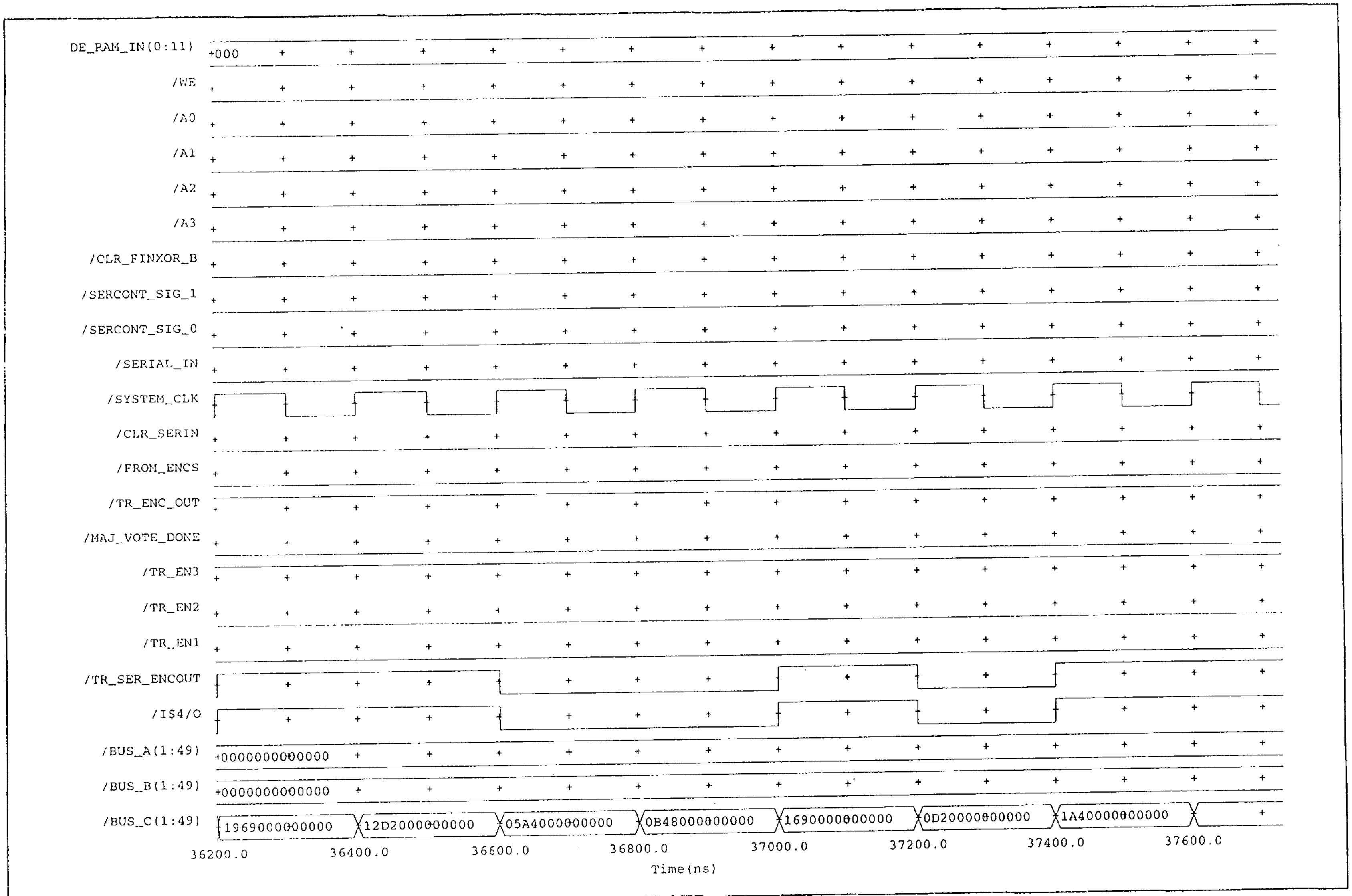

Figure B.3. NEWTEST_YEN Registers Shift End Trace.

```
   0.0  *1  *0r  *Xr  *XXXXXXXXXXXXXr  *XXXXXXXXXXXXXr  *XXXXXXXXXXXXXr
   0.1   1   0r   Xr   XXXXXXXXXXXXXr   XXXXXXXXXXXXXr   XXXXXXXXXXXXXr
   0.2   1   0r   Xr   XXXXXXXXXXXXXr   XXXXXXXXXXXXXr   XXXXXXXXXXXXXr
 100.0  *0  *1   *0   *0000000000000   *0000000000000   *0000000000000
 200.0  *1  *0    0    0000000000000    0000000000000    0000000000000
 300.0  *0   0    0    0000000000000    0000000000000    0000000000000
 400.0  *1   0    0    0000000000000    0000000000000    0000000000000
 500.0  *0   0    0    0000000000000    0000000000000    0000000000000
 600.0  *1   0    0    0000000000000    0000000000000    0000000000000
 700.0  *0   0    0    0000000000000    0000000000000    0000000000000
 800.0  *1   0    0    0000000000000    0000000000000    0000000000000
 900.0  *0   0    0    0000000000000    0000000000000    0000000000000
1000.0  *1   0    0    0000000000000    0000000000000    0000000000000
1100.0  *0   0    0    0000000000000    0000000000000    0000000000000
1200.0  *1   0    0    0000000000000    0000000000000    0000000000000
1300.0  *0   0    0    0000000000000    0000000000000    0000000000000
1400.0  *1   0    0    0000000000000    0000000000000    0000000000000
1500.0  *0   0    0    0000000000000    0000000000000    0000000000000
1600.0  *1   0    0    0000000000000    0000000000000    0000000000000
1700.0  *0   0    0    0000000000000    0000000000000    0000000000000
1800.0  *1   0    0    0000000000000    0000000000000    0000000000000
1900.0  *0   0    0    0000000000000    0000000000000    0000000000000
2000.0  *1   0    0    0000000000000    0000000000000    0000000000000
2100.0  *0   0    0    0000000000000    0000000000000    0000000000000
2200.0  *1   0    0    0000000000000    0000000000000    0000000000000
2300.0  *0   0    0    0000000000000    0000000000000    0000000000000
2400.0  *1   0    0    0000000000000    0000000000000    0000000000000
2500.0  *0   0    0    0000000000000    0000000000000    0000000000000
2600.0  *1   0    0    0000000000000    0000000000000    0000000000000
2700.0  *0   0    0    0000000000000    0000000000000    0000000000000
2800.0  *1   0    0    0000000000000    0000000000000    0000000000000
2900.0  *0   0    0    0000000000000    0000000000000    0000000000000
3000.0  *1   0    0    0000000000000    0000000000000    0000000000000
3100.0  *0   0    0    0000000000000    0000000000000    0000000000000
3200.0  *1   0    0    0000000000000    0000000000000    0000000000000
3300.0  *0   0    0    0000000000000    0000000000000    0000000000000
3400.0  *1   0    0    0000000000000    0000000000000    0000000000000
3500.0  *0   0    0    0000000000000    0000000000000    0000000000000
3600.0  *1   0    0    0000000000000    0000000000000    0000000000000
3700.0  *0   0    0    0000000000000    0000000000000    0000000000000
3800.0  *1   0   *1   *1EBC98234570A   *1A234598BCDEA   *189DCA3EF5870
3800.3   1   0    1    1EBC98234570A    1A234598BCDEA    189DCA3EF5870
3900.0  *0   0    1    1EBC98234570A    1A234598BCDEA    189DCA3EF5870
4000.0  *1   0   *0   *0000000000000   *0000000000000   *0000000000000
4100.0  *0   0    0    0000000000000    0000000000000    0000000000000
4200.0  *1   0    0    0000000000000    0000000000000    0000000000000
4200.1   1   0    0    0000000000000    0000000000000    0000000000000
4300.0  *0   0    0    0000000000000    0000000000000    0000000000000
4400.0  *1   0    0    0000000000000    0000000000000    0000000000000
Time(ns)  ^/SYSTEM_CLK                  ^/BUS_B(1:49)    ^/BUS_C(1:49)
              ^/MAJ_CLR
                  ^/MAJ_VOTE_EN
                      ^/BUS_A(1:49)
```

Figure B.4. DECODE_STAGE_LAST2 Simulation Inputs List.

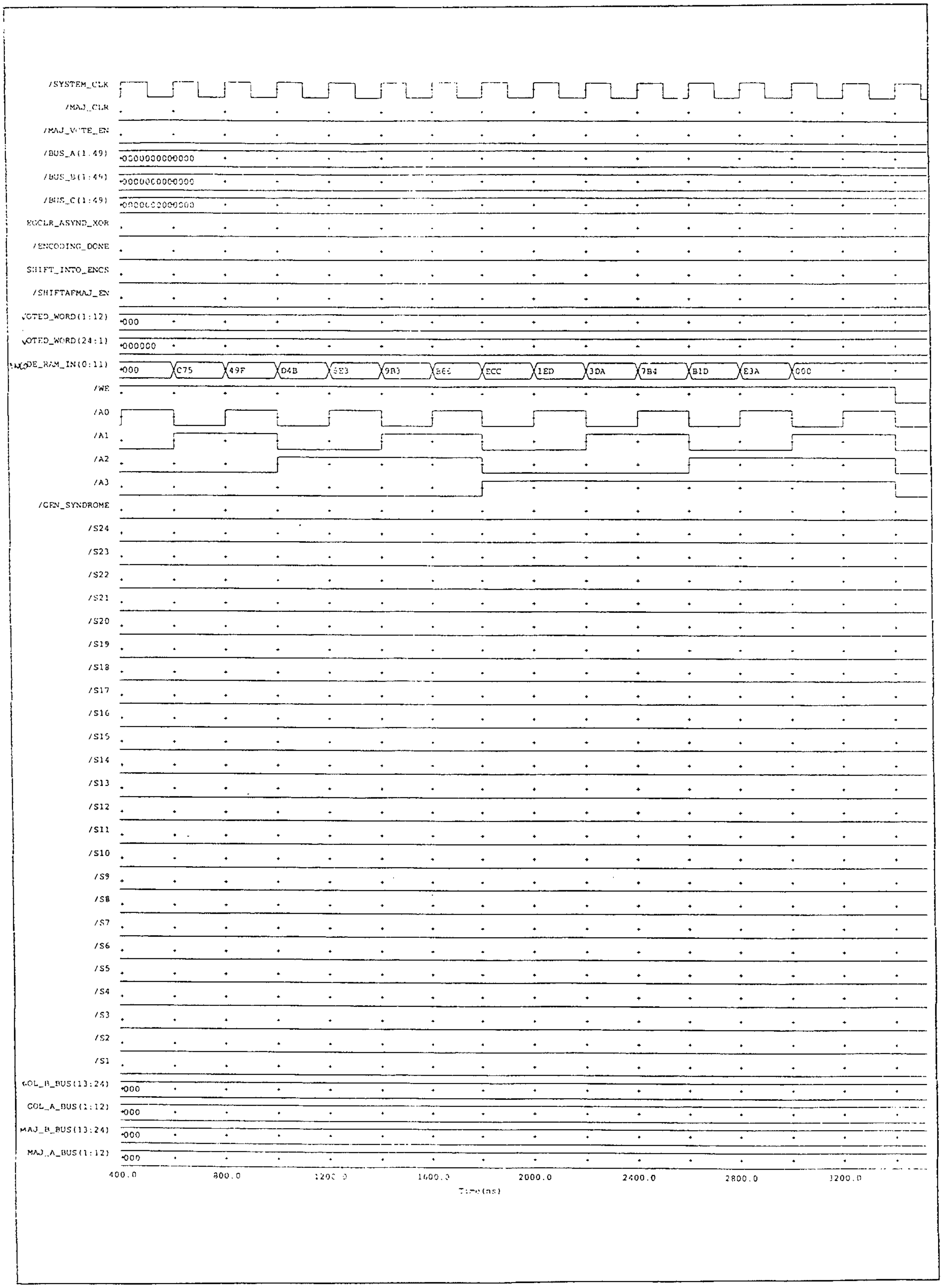

Figure B.5. DECODE_STAGE_LAST2 Encode ROM Trace.

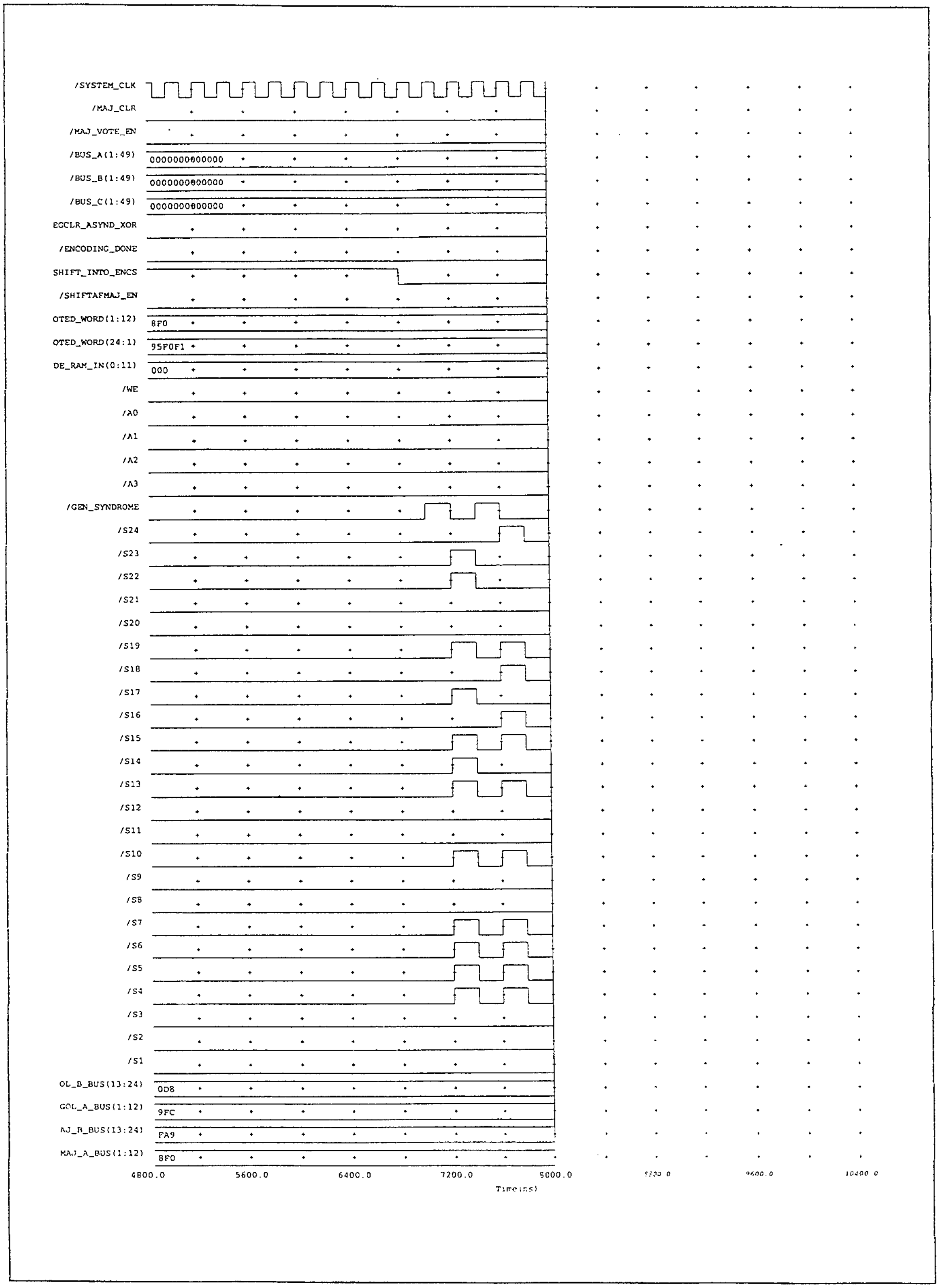

Figure B.6. DECODE_STAGE_LAST2 Syndrome Bits Trace.

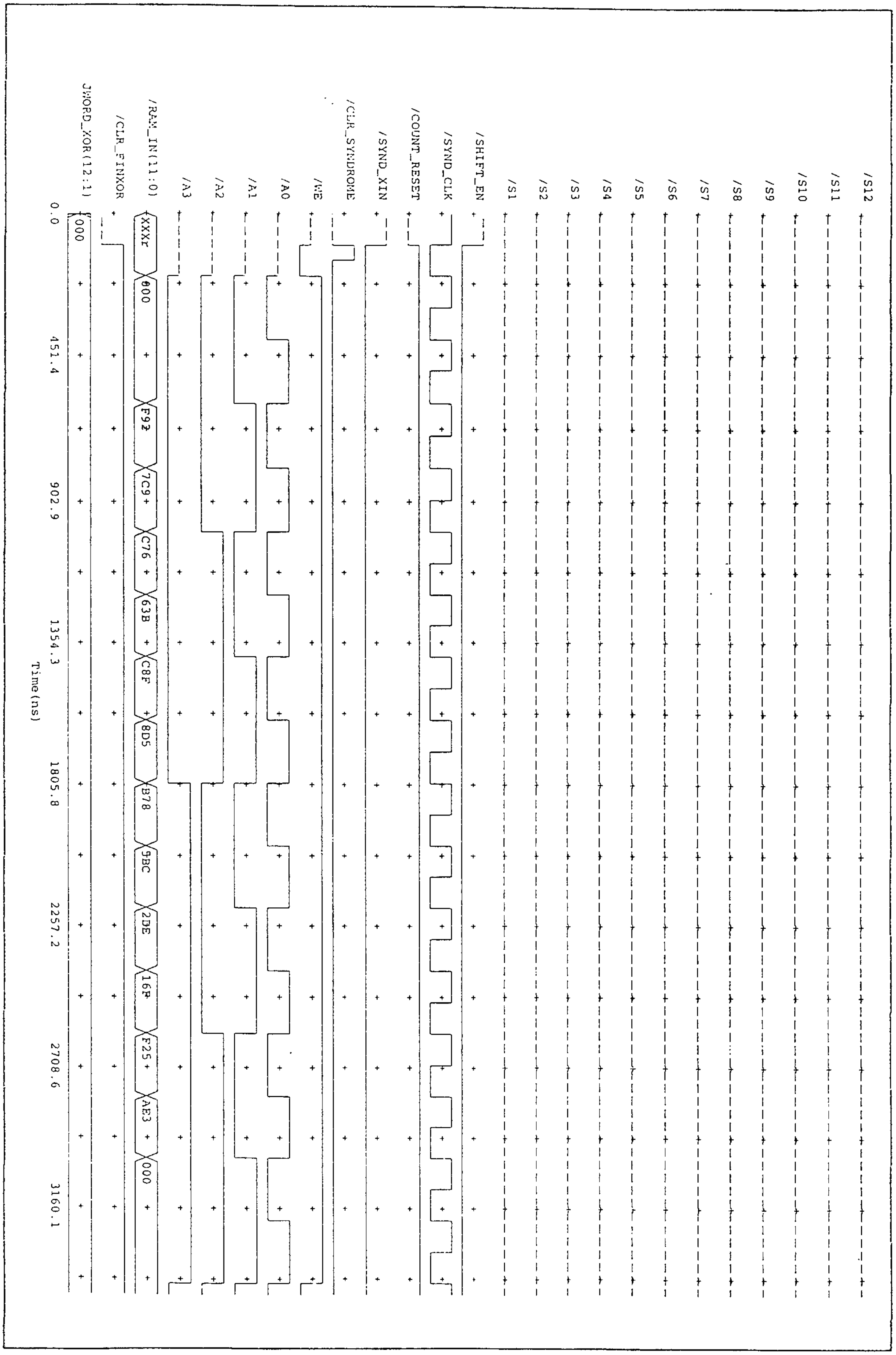

Figure B.7. RAM_BITS_NEW1 Decode ROM Trace.

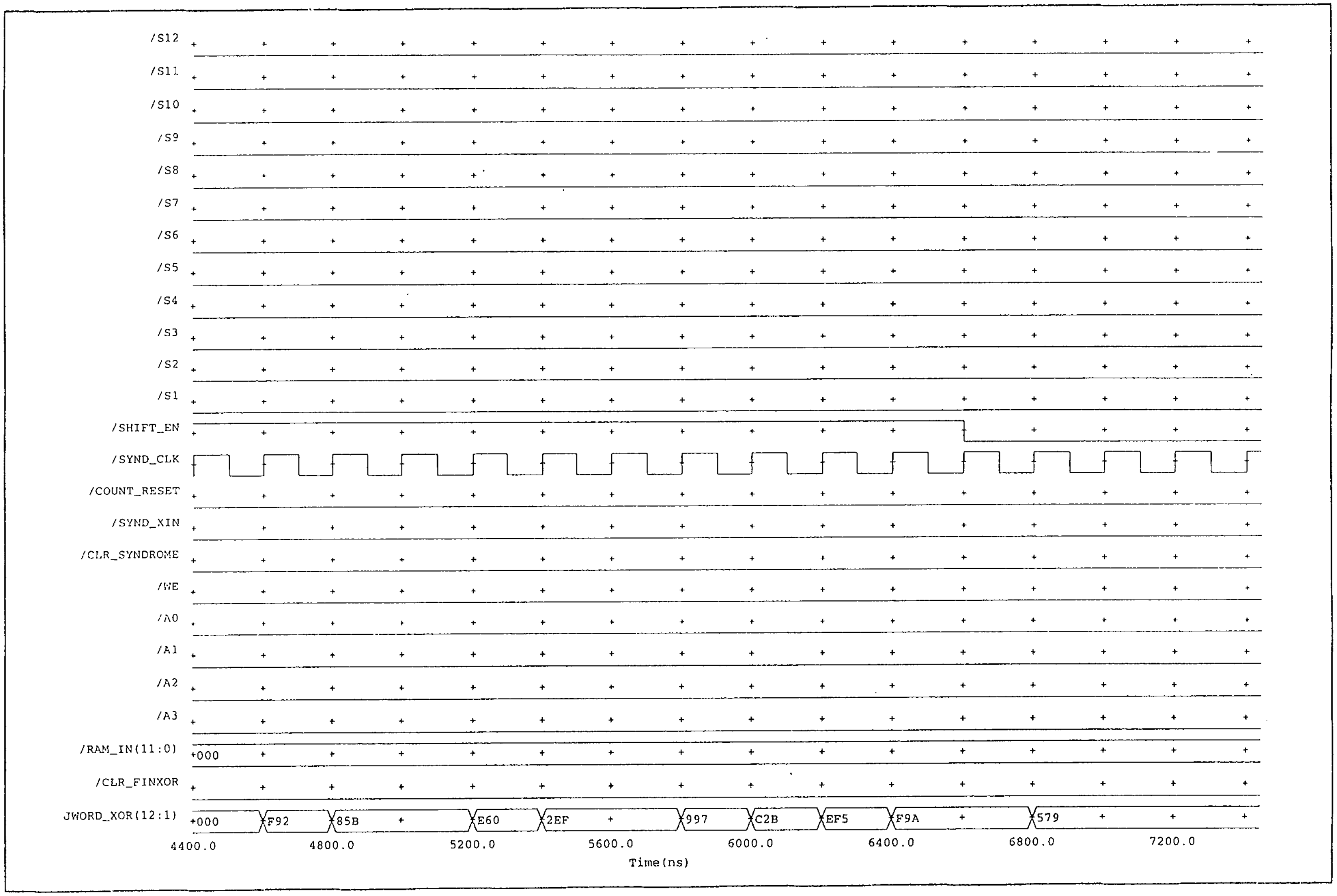

Figure B.8. RAM_BITS_NEW1 Data Bits Error Pattern to Correct Trace.

LIST OF REFERENCES

John F. Wakerley, *Digital Design Principles & Practices*, Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1994.

Mentor Graphics Inc., *Idea Station, Personal Learning Program*, 1989.

Proakis John G., *Digital Communication*, Mc Graw Hill, Inc, New York, NY, 1989.

Shu Lin, Daniel J. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice Hall, Inc, Englewood Cliffs, New Jersey, 1983.

Snelgrove, Andrew H., "The Design of Programmable Convolutional Encoder Using VHDL and an FPGA," December 1994.

William Stalling, *Data and Computer Communications*, McMillan Publishing Company, New York, NY, 1994.

XILINX, Inc, *XC3000and XC4000 Family Programmable Gate Array Training Course Day1, Day2, Day3, Day4*, San Jose, CA, 1995.

XILINX, Inc, *Programmable Logic Breakthrough'95*, San Jose, CA, 1995.

XILINIX, Inc, *The Programmable Logic Data Book*, San Jose, CA, 1994.

XILINX, Inc, *User Guide and Tutorials*, San Jose, CA, 1991.

XILINX, Inc., *XACT Libraries Guide*, San Jose, CA, 1994.

XILINX,Inc., *XACT MENTOR Version & Interface User Guide*, San Jose, CA, 1994.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center 8725 John J. Kingman Rd., STE 0944 Ft. Belvoir, VA 22060-6218	2
2.	Dudley Knox Library Naval Postgraduate School 411 Dyer Rd. Monterey, CA 93943-5101	2
3.	Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
4.	Professor Chin-Hwa Lee, Code EC/Le Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
5.	Professor Todd Weatherford, Code EC/We Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5121	1
6.	Deniz Kuvvetleri Komutanligi Personel Daire Baskanligi Bakanliklar, Ankara, Turkey	2
7.	Deniz Harp Okulu Komutligi Tuzla, Istanbul, Turkey 81704	1
8.	Mehmet Sari Dz.Utgm. Deniz Kuvvetleri Komutanligi Muhabere Eln. Bsk.'ligi Sahil Sistemleri Gelistirme Proje Sb. Bakanliklar, Ankara, Turkey	1

9.	Istanbul Teknik Üniversitesi Universite Kütüphanesi Maslak, Istanbul, Turkey	1
10	Orta Dogu Teknik Universitesi Universite Kutuphanesi Balgat, Ankara, Turkey	1
11.	Golcuk Tersanesi Komutanligi APGE Baskanligi Golcuk, Kocaeli, Turkey	1
12.	Bogazici Universitesi Universite Kutuphanesi Bebek, Istanbul, Turkey	1