Reliable Data Processor in VLSI

EE 452 Final Report
Senior Capstone Project

By
Rahul Chopra

Advisor: Dr. Vinod Prasad

Overview:

1. **Abstract**
2. **Introduction and Block Diagram**
3. **Functional Description**
4. **Subsystems**
5. **Logic Works/VLSI Implementation of Subsystems, P-Spice Test Results**
6. **VHDL Implementation and Output**
7. **Conclusions**
8. **Timeline**
9. **Date Sheets & References**

Abstract:
The Reliable Data Processor in VLSI is a user driven data processor that carries out Arithmetic and Logic operations such as AND, OR, XOR, NAND and ADD on 4-bit data. The basic system consists of a 4-bit ALU controlled by a user driven controller. The Reliable Data Processor also includes an encoder to encode the output using Hamming code and registers to save the output in. The system has feedback capabilities and receives feedback from an external reliable chip indicating any erroneous data transfer upon which it resends data. Applications of the data processor include Digital Signal Processing and data encryption. The system was designed and simulated at the gate level using Logic works after which it was built in VLSI using CMOS scaleable technology utilizing the software program L-Edit. The design was further verified using VHDL code written and then implemented on the Xilinx XC4005E FPGA board.

Introduction:

This project utilizes the design techniques and fundamentals learned in classes such as computer architecture, reliability, and VLSI Design to build a 4-bit Arithmetic and Logic Unit. These principles enabled the design of an Arithmetic and Logic Unit that accepts feedback from an error detection chip, thus reducing data transfer errors. In addition to this, the system also encodes the ALU output using the hamming code. This makes it compatible with many digital systems, as the hamming code has been a prevalent standard since the 1930's. This system has applications in Digital Signal Processing and Data Encryption systems, with reliability feedback capabilities rare in existent systems in these areas.

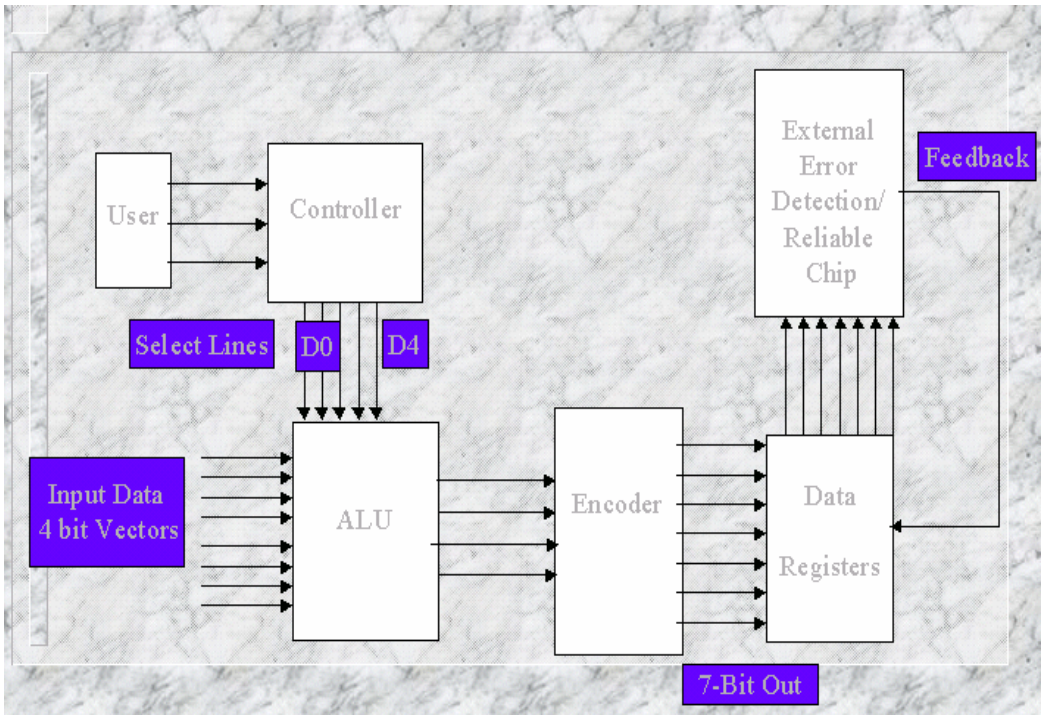

Figure 1-1 Complete System Block Diagram

Functional Description:

The main inputs to the chip consist of two 4-bit vector arrays, which supply the ALU with input data. In addition to this, the system accepts a 3-bit user input, which indicates the users choice. Based on this choice, the controller instructs the ALU to perform one of 5 different arithmetic and logic operations, which are, AND, OR, XOR, NAND and Addition on the two 4-bit vector arrays. The ALU on performing operations on these 4-bit vectors produces a 4-bit output vector and a carry flag to represent the carry bit output for the add function of the ALU. The 4-bit output of the ALU is fed into the encoder subsystem, which uses hamming code to generate parity bits and encode the output. The output is then stored in data registers and simultaneously sent to an external error detection circuitry for reliability testing. The external chip checks for erroneous data transfer using a hamming window technique and sends back an output to the register subsystem indicating so.

Inputs	**Description**
Vector A	4-bit vector array
Vector B	4-bit vector array
User Inputs- A0, A1, A2	Select Operations to be performed. (A0-MSB, A2-LSB)
Error Signal	Signal from reliable chip indicating error.
Outputs	
Encoder/Register Outputs	7-bit output to external reliable/error correction chip.

Table 4-1 I/O Table for Complete System

Subsystems:

Controller:

The controller controls the ALU. The input to the controller is the users choice designated by a 3-bit input. The controller offers the user a choice of arithmetic and logic operations and based on what the users picks, the controller instructs the ALU to carry out those operations.

ALU:

The ALU or the Arithmetic and Logic Unit performs basic arithmetic and logical operations on input data. These include AND, OR, NAND, XOR and ADD operations. The ALU receives input data in the form of two 4-bit vector arrays along with select lines from the controller which contain the decoded user instructions instructing the ALU on what operations to carry out. Upon carrying out the operations, the ALU forwards the 4-bit output to the encoder.

Encoder:

The encoder receives the 4-bit output from the ALU and adds 3 parity bits to output a 7-bit data vector. This output is then sent to the external reliable chip for testing and also to the registers for data storage.

Register Subsystem:

The register subsystem consists of data registers in the form of D-Flip Flops. The data registers store the output from the encoder in 7 D-Flip Flops the clock input of which is the signal from the error detection chip. This clock input controls the D-Flip Flops, thus effectively introducing feedback into the system. In case there is erroneous data transfer to the external chip, the output stored in the Data Registers will be resent to the external chip.

Approach/Implementation:

The system was first built and designed in Logic works. During this process, each subsystem was individually designed, built, and tested at the gate level. After a subsystem was built and tested in Logic works, the design was implemented in VLSI using the L-Edit simulation package. The completed L-Edit designs of each subsystem were tested as a stand-alone by observing outputs using Microsim's P-Spice software after which all subsystems were integrated and tested as one L-Edit file. The design was also programmed in VHDL code, and this VHDL code was implemented in FPGA using a Xilinx XC4005 FPGA board and the Xilinx software.

Logic Works/VLSI Implementation:

Controller:

The first subsystem is the controller. Figure 7-1 represents the decoding logic, with A0, A1 and A2 representing the user inputs to the controller, and D0-D4 representing the select line outputs to the ALU instructing it to carry out various arithmetic/logic operations.

A0	A1	A2	D0	D1	D2	D3	D4
0	0	0	1	0	0	0	0
0	0	1	0	1	0	0	0
0	1	0	0	0	1	0	0
0	1	1	0	0	0	1	0
1	0	0	0	0	0	0	1
OPERATION			**AND**	**OR**	**XOR**	**NAND**	**ADD**

Table 5-1, Decoding Logic Table

If the user inputs a 010 as his 3-bit choice representing A0, A1 and A2, the controller will decode it to indicate D1 showing a 5V or logic "1" signal, thus instructing the ALU to perform an OR operation on the two 4-bit vectors. This and other combinations for table 7-1 are shown in the P-Spice simulation outputs of the controller in figure 8-3. Figure 8-1 shows the NAND Implementation of the 3*8 decoder used for the controller in logic works. Figure 8-2 is the VLSI Implementation in L-Edit of the same circuit.

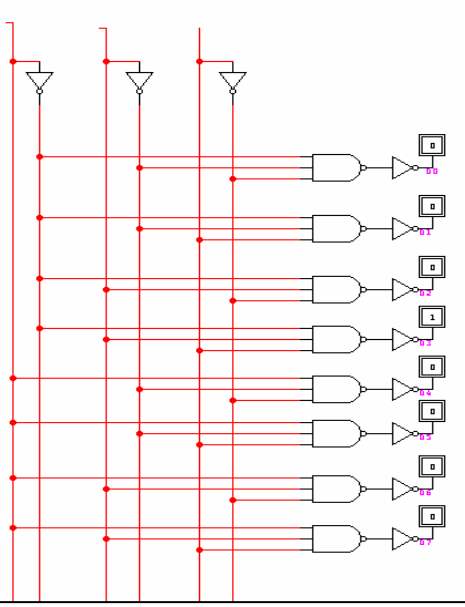

Fig. 6-1 Logic Works Design of Controller

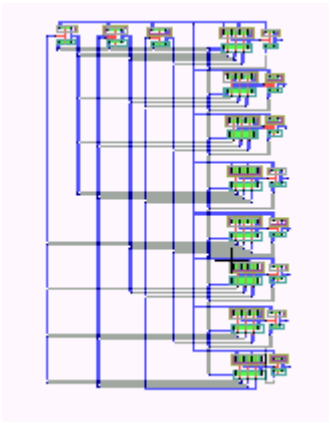

Fig. 6-2 L-Edit Design of Controller

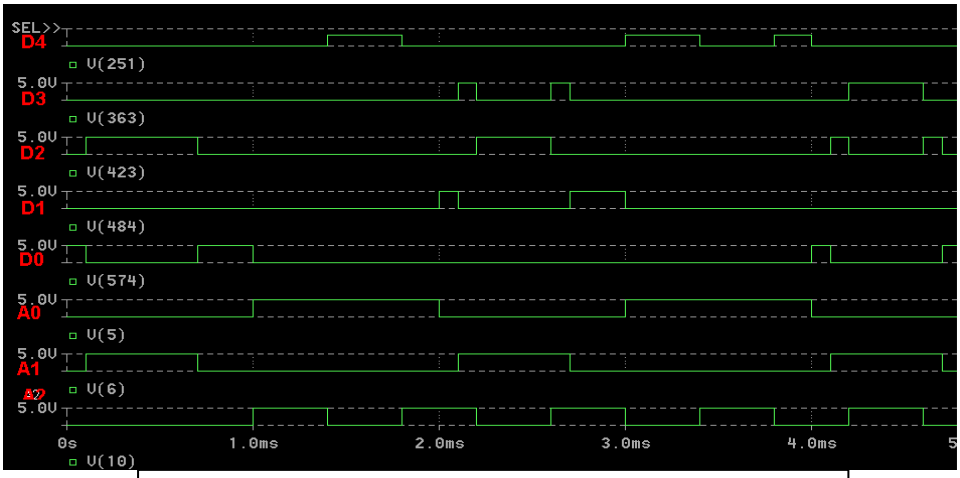

Fig. 6-3 P-Spice Output Showing Controller Output

Registers:

The register subsystem consists of 7 D-Flip Flops that store the encoded output from the encoder. Figure 9-1 shows the logic works design and timing analysis of the D-Flip Flop. The D-Flip Flop has set and reset capabilities with set having higher priority. All 7 D-Flip Flops in the register subsystem are connected together at the clock input with the feedback signal from the external reliable chip. Thus, this signal in affect drives the registers, and they hold data and send it according to this signal. The flip flops are falling edge triggered, thus, when the signal from the error detection chip changes from 1 to 0, all 7 D-Flip Flops re-send the data bit stored in them to the external reliable chip as this indicates erroneous data transfer. When the Set-Bar input is low, the flip-flops set to 1, and when the Reset-bar is low, they reset the information stored in them to 0. In case both are low, Set-bar has a higher priority. Figure 10-1 shows the same results in P-Spice for a single D-Flip Flop.

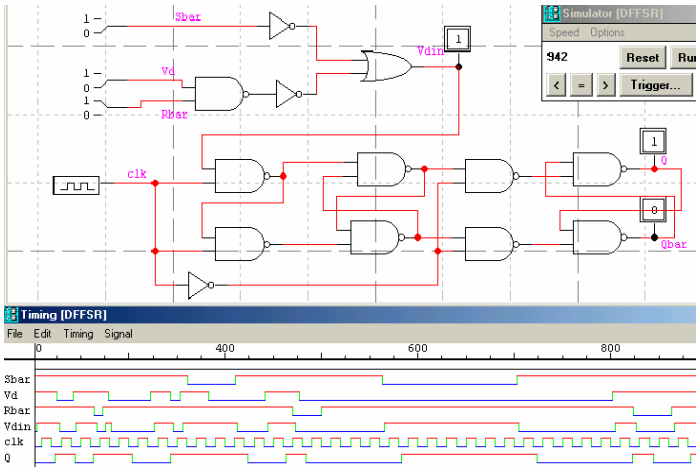

Fig. 7-1, Logic Works and Timing of D-FF

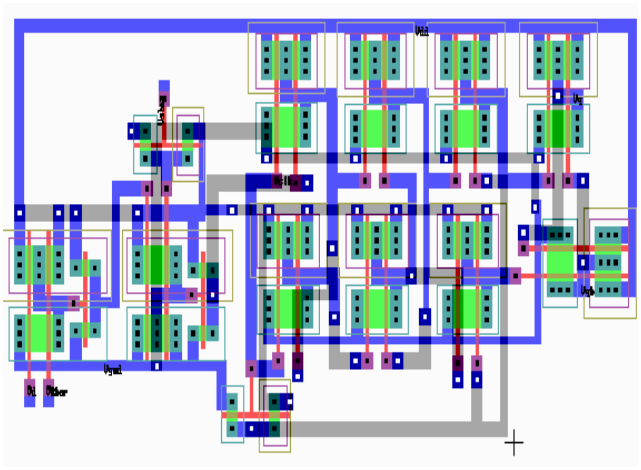

Fig. 7-2, L-Edit Design of D-FF

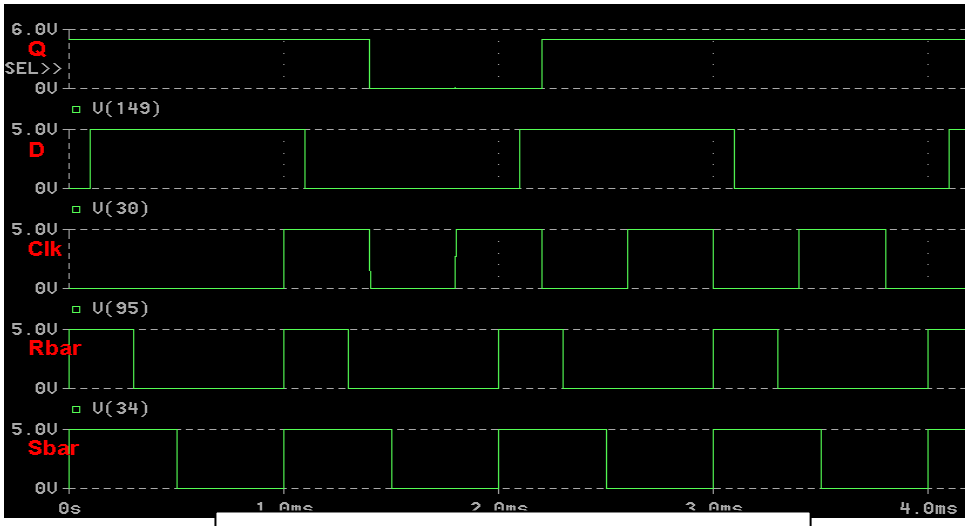

Fig. 7-3, P-Spice Timing Results for D-FF

Encoder:

The encoder accepts the 4-bit output from the ALU and adds 3 parity bits using the hamming code. The following three equations determine the 3 parity bits P4, P2 and P1 with I7, I6, I5 and I3 being the 4 information bits.

P1= I3(+)I5(+)I7
P2= I3(+)I6(+)I7
P4= I5(+) I6(+)I7

The output 7 bits from the Encoder are fed into the Data Register subsystem, and at the same time they go to the external reliable chip.
Figure 10-1 shows the encoder in Logic Works Design, and Figure 10-2 shows the L-Edit design.

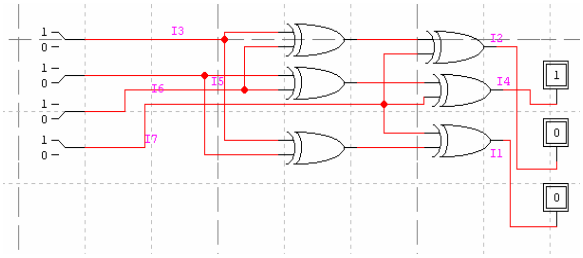

Fig. 8-1 Logic Works Design of Encoder

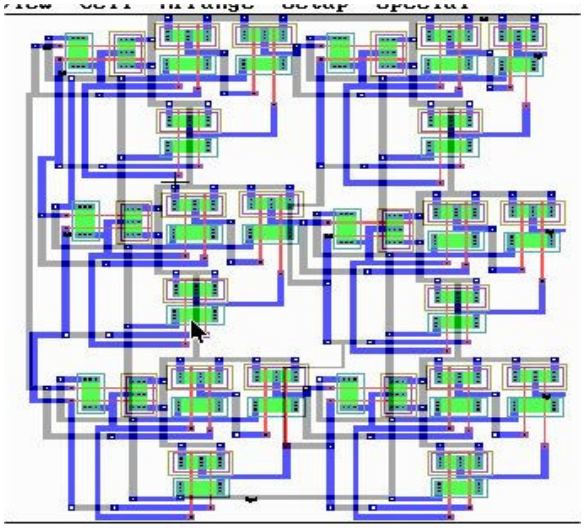

Fig. 8-2 L-Edit Design of Encoder

ALU:

The ALU was design consisted of 5 different subsystems, each doing a certain arithmetic/logic operation on input data. The first ALU operation is the AND operation. For this, 4 3-Input AND Gates were built, with two inputs being the respective bits from the vectors, and the third input coming from the select line. This insured that the output would only be enabled when the select line D0 was high. Similarly for the OR operation of the ALU, two input OR gates were built in L-Edit, and the output of the OR gates was ANDED with the select line D1, thus ensuring that the OR outputs would be activated only when the user chose to perform an OR operation. The Logic works simulation and results are shown below for the AND operation, along with L-Edit design of the AND, and OR functions.

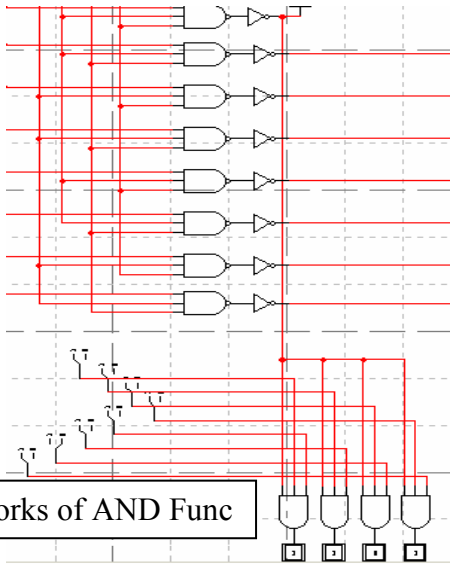

Fig.8-3 Logic Works of AND Func

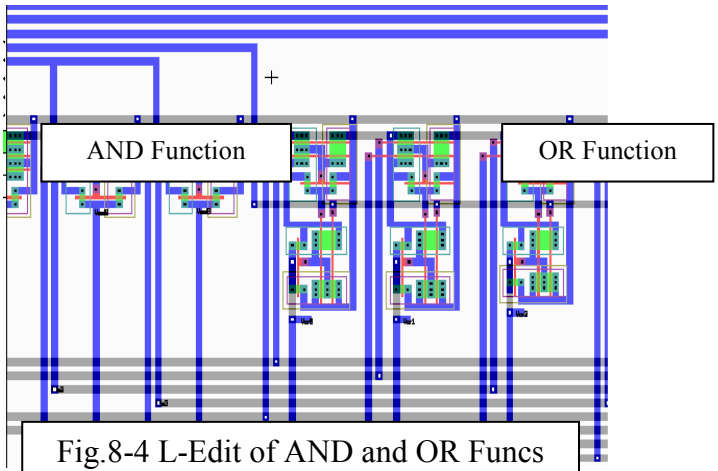

Fig.8-4 L-Edit of AND and OR Funcs

The XOR function was built using a combination of NAND Gates. For the NAND function, simple 2 input NAND gates were used. The outputs for both these operations were NANDED with their respective select lines to ensure that these operations forward results to the encoder only when the user selects these operations. The L-Edit designs of the XOR and NAND functions of the ALU are shown below.

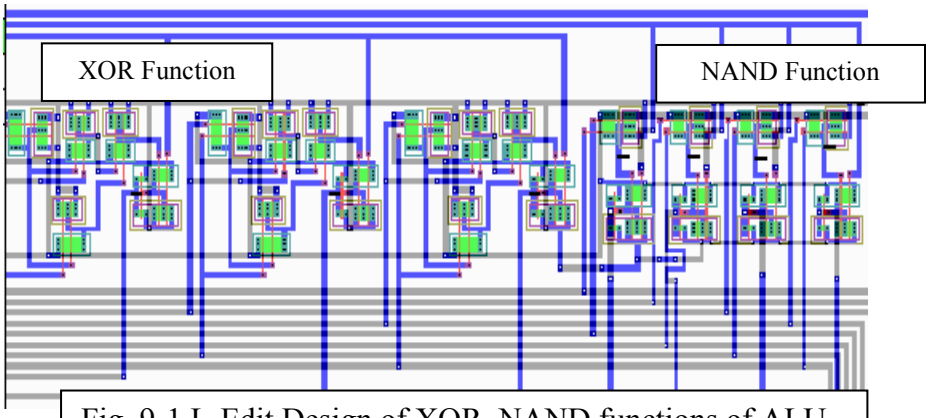

Fig. 9-1 L-Edit Design of XOR, NAND functions of ALU

The adder system performs the addition of two 4-bit vectors. It is built using 2-Input XOR, AND, and OR gates. It utilizes a full adder design, the basic difference being that it feeds the carry out of one addition into the next. It creates a 4-bit output like all other operations of the ALU. It also sets the carry flag according to the carry output of the addition. The carry flag is only affected by add operations and is not sent out to the encoder as part of the output.

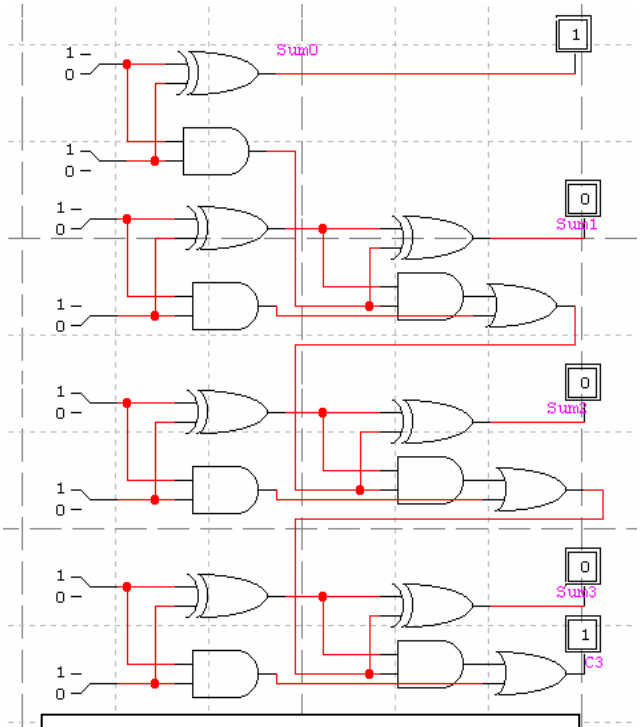

Fig. 9-2 Adder Circuit-Logic Works

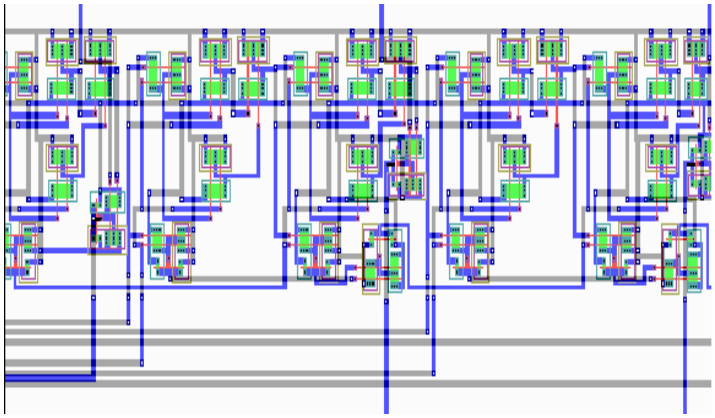

Fig. 9-3 Adder Circuit-L-Edit Design

The complete L-Edit design of the Reliable Data Processor in VLSI is shown below with the different subsystems marked.

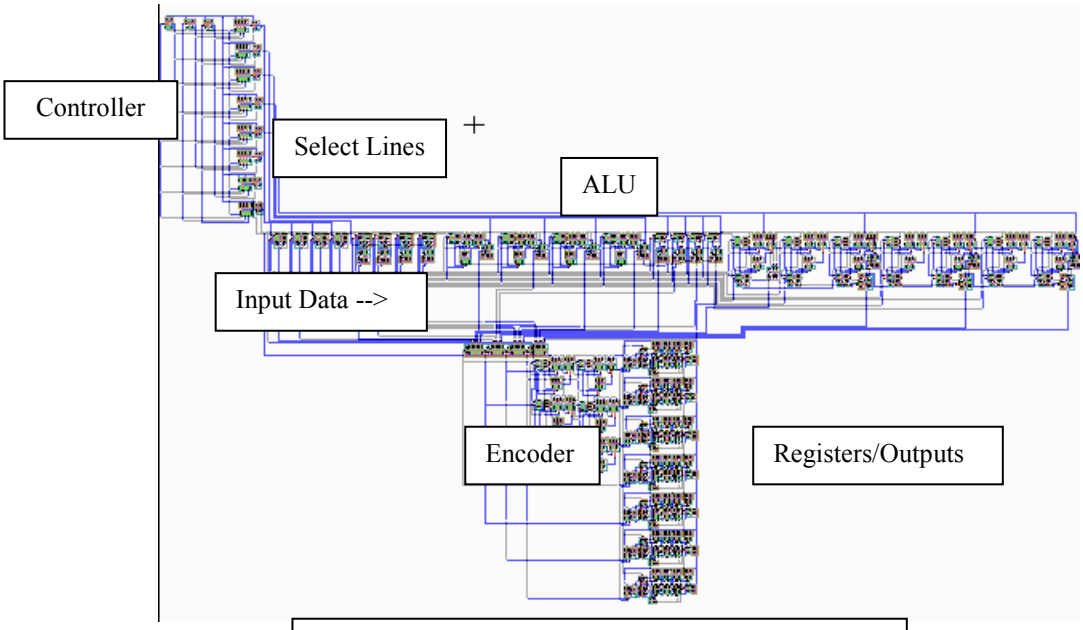

Fig. 10-1 Completed L-Edit Design

The P-Spice results from the ALU are shown below. Figure 10-2 shows an ADD function being implemented, which shows the select line D4 being high. For input data on the vectors reading 1011 and 1111 respectively, the ADD function results show the output to be 1010 and shows the carry flag as 1.

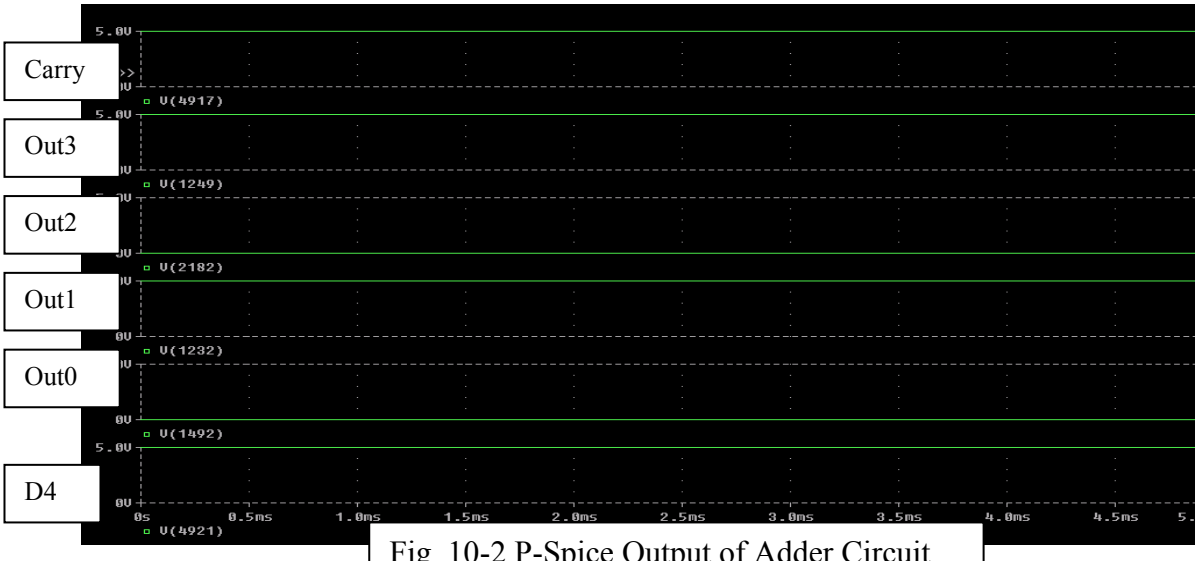

Fig. 10-2 P-Spice Output of Adder Circuit

The output shown in figure 10-2 is encoded, and the encoded output is shown in figure 11-1. For an input of 1010 to the encoder, it gives an output of 1010010 using the hamming code.

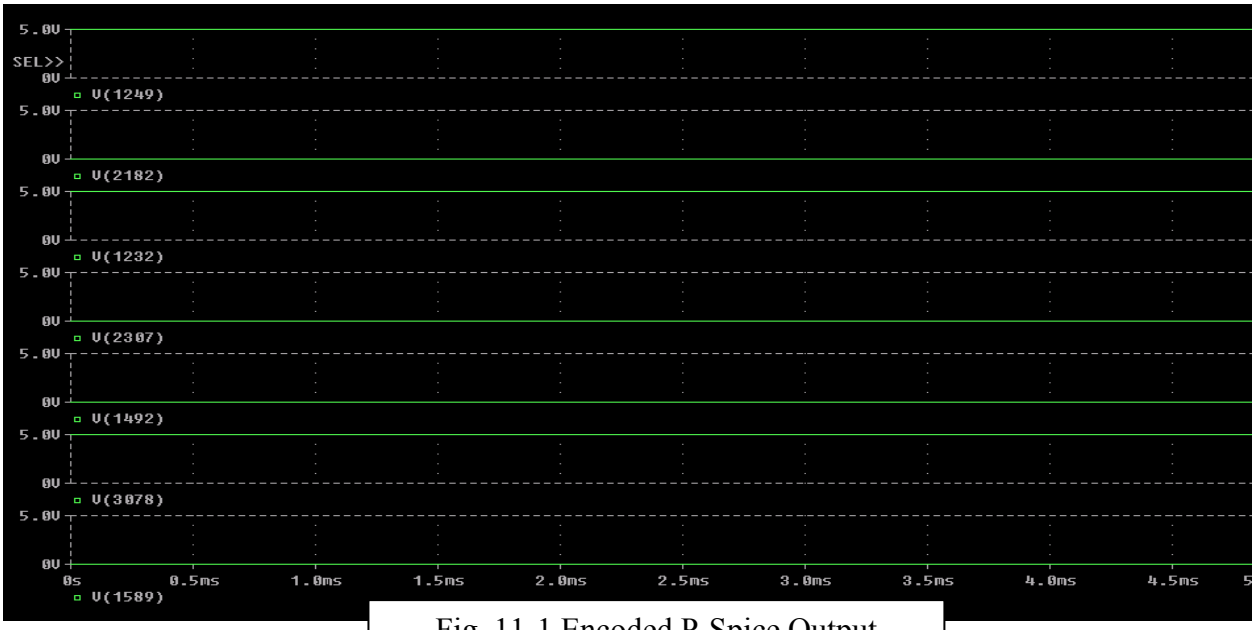

Fig. 11-1 Encoded P-Spice Output

VHDL/FPGA Implementation:

For the VHDL/FPGA implementation, code was written for the logic section of the ALU to show NAND, NOR, XOR and AND operations based on the select lines from a 2*4 decoder. This code was implemented on an Xilinx XC4005E FPGA chip and a working data processor was displayed at the student expo on April 12th 2002. Figure 11-2 shows the testing results. The 2 bit vector ab represents the user input to the decoder, X and Y represent two 4-bit vectors, and Z represents the output from the ALU. The 7 bit vector encout represents the encoded output.

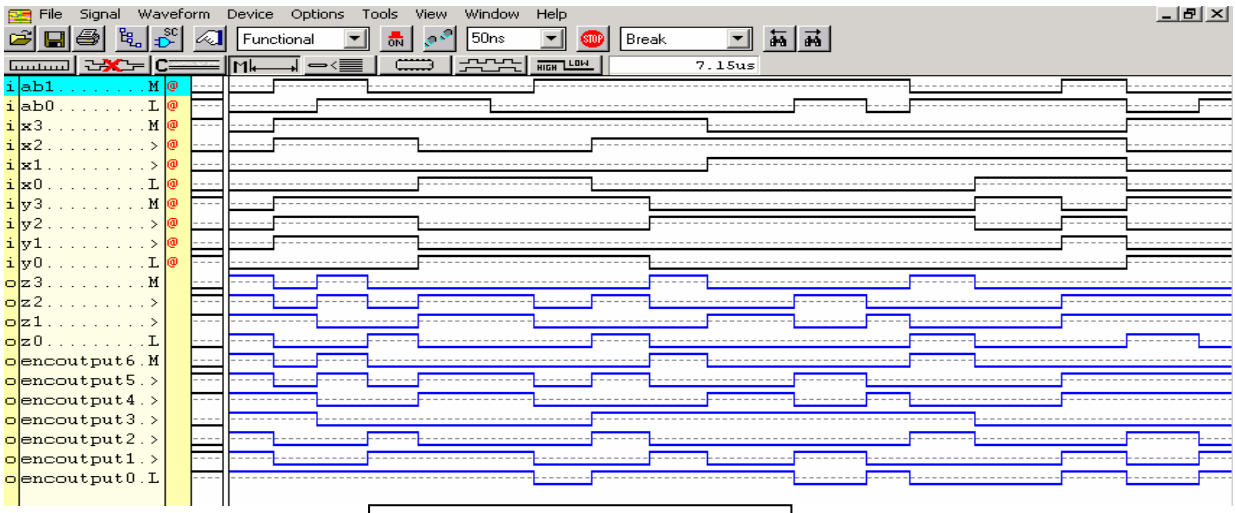

Fig. 11-2 VHDL Code

Conclusions: