DESIGN, MANUFACTURE AND TESTING OF A 4-BIT MICROPROCESSOR

Theodore D'Antonoli
5th Year Microelectronic Engineering Student
Rochester Institute of Technology

ABSTRACT

A four bit microprocessor was designed using an Apollo workstation and MOSIS two lambda design rules. The layout was intended for fabrication with a four level, enhancement mode load, NMOS process. Device operation was verified by breadboarding the microprocessor with 74'00 series CMOS logic parts. Testing of the breadboard showed that all functions operated correctly. However, the final breadboard design indicated the need for revision of the original layout.

INTRODUCTION

At the heart of every computer is a microprocessor; a device composed of modules such as counters, adders, registers, and tristate buffers. Proper connection of these elements enables the device to perform desired operations on binary numbers, such as addition, subtraction, and Boolean logic. The microprocessor will perform a specific operation when it receives a binary coded instruction. Complex routines, which are derived from more basic operations, are performed via a series of instructions. This requires the addition of a memory device (i.e. static or dynamic RAM) to store instruction sequences and data.

The design of a microprocessor begins by determining the specific operations the device is required to perform. Modules are then designed and interconnected to perform these operations. The interaction between modules and memory indicates the architecture of the microprocessor. Two of the more popular architectures are Harvard and von Neumann. The basic difference between these architectures is the manner in which programs and data are stored. The Harvard architecture contains one memory to store programs and a second memory to store data. High speed devices are possible with this architecture because coprocessing operations may be utilized. Devices which mix programs and data together in the same memory are called von Neumann devices [1]. This allows for more efficient memory allocation. While this may result in a slower device, the design becomes much simpler, since only a single memory is required. Use of a single memory requires a bus which transfers data and address information throughout the device. All modules are connected to the bus and interface logic determines whether or not it will see the data on the bus. All modules are also free to place data onto the bus.

DESIGN

This project involved the design of a 4-bit microprocessor which was called the MAVERICK device. Von Neumann architecture was chosen to implement the instruction set shown in Table 1. Implementation of the instruction set required the design of several modules including a 4-bit full-adder, registers, tristate buffers and a control unit (decoder) to direct the flow of data through the device. The device will add, subtract, increment, decrement, NAND and NOR two 4-bit binary numbers, and INV a 4-bit binary number. The load (LDA) and store (STA) instructions are intended for use with a 256 x 8 byte, random access memory (RAM) which is external to the microprocessor. The lower 224 memory locations are allocated for instructions and data which may be read into the microprocessor. The upper 32 memory locations contain only data which can be accessed with load and store instructions.

Table 1: Instruction Set For MAVERICK Device

OP code	Function	Description
0000	<A> - 1	Decrement <A> by 1
0001	<A> + 1	Increment <A> by 1
0010	<A> - <B>	Subtract <B> from <A>
0011	<A> + <B>	Add <B> to <A>
0100	LDA	Load ACC with data from lower 4 bits of MDR
0101	Unused	
0110	Unused	
0111	HALT	Stop all operations
1000	NAND <A> and <B>	Bit by bit NAND of <A> with <B>
1001	NOR <A> and <B>	Bit by bit NOR of <A> with <B>
1010	INV <A>	Complement <A>
1011	NOP	Increment the program counter (PC)
110X	STA	Store <A> in address given by lower 5 bits on MDR
111X	LDA	Load ACC with data given by lower 4 bits of MDR

MDR = Memory Data Register
ACC = Accumulator
<A> = Contents of the accumulator (4 bit binary number)
<B> = Data on the lower 4 bits of the memory data register

The steps necessary to execute an instruction are:

1.) Fetch instruction from memory location held in a program counter (PC)
2.) Decode operation
3.) Perform Operation and Update the PC

This sequence requires a two-phase, non-overlapping clock (referred to as clk1 and clk2) with voltage levels of 5 volts when on and 0 volts when off. During a clk1 pulse (5v), the address in the program counter (PC) is placed in the memory address register (MAR), the READ line of the memory device is enabled, and an instruction is placed in the memory data register (MDR). This corresponds to the fetch instruction operation. The clk2 pulse then decodes the instruction which generates enable signals responsible for instruction execution and updates the PC. The cycle is then repeated. Therefore, during each clock pulse, a number of different operations are performed at the same time. This eliminates concerns regarding timing based on rise and fall times. Other timing issues based on device clocking may be resolved by lengthening the duration of the clock. However, this is not possible when dealing with the operation of the WRITE line. If this line is enabled before the address lines fully charge, data may be written into a number of memory locations while an address is placed on the address bus. A non-inverting chain of inverters was used to delay the signal on the WRITE enable line. If this delay does not prove to be sufficient, external delay may be added. Other timing issues may be resolved by lengthening the duration of each clock pulse.

An instruction consists of an eight bit word. The upper four bits contain a binary coded instruction while the lower four bits contain data. For example, the eight bit word, 01001001, will load (0100) the binary number 1001 into the accumulator. This configuration was altered only for addressable load and store operations. In these cases, the upper three bits contain the instruction while the lower five bits contain address information. Any address indicated for these instructions refers to one of the upper 32 locations in memory. The X's in Table 1 for these instructions indicate the additional bit of the instruction, used for address information. An example would be to store the accumulator contents into memory location 21 (actually memory location 245). The instruction for this is 11010101. The upper 3 bits (110) indicate an addressable load is to be performed while the lower 5 bits provide address information. The address which will be placed on the address bus will be 11110101. The upper 3 bits of an address will always be 111 to indicate a location in upper memory. This is also true of the STA instruction.

The interface logic contained in various modules consists of a series of non-inverting, tristate buffers which allow data to be placed onto the bus. The registers used to store data consist of a series of transparent, gated D flip flops. Any data reaching the input of a register during an enable pulse will be latched. The program counter consists of a series of eight D flip flops, connected so that clk2 will cause the output to increment by 1. The device is capable of counting from 0 - 256. The arithmatic logic unit (ALU) consist of some logic modules and an adder subtractor unit. The logic modules perform a bit by bit logic operation on the contents in the accumulator. An example of a logic function is the NOR of 1100 with 0110. The

accumulator contains one of the binary numbers (say 1100). The second binary number is contained in the lower four bits of the MDR. The instruction for the NOR operation of the accumulator with 0110 is then 10010110. The ACC will contain 0001 after after the instruction has been executed.

The arithmatic unit is composed of an adder/subtractor module and logic modules. The adder/subtractor module consists of a standard 4-bit ripple, full adder. Some simple input logic was added to realize the increment, decrement and subtract functions. Logic modules consist of a series of NOR gates, inverters and NAND gates to perform bit by bit Boolean logic operations. All circuit schematics can be found in the Appendix in Figures A1 - A7.

LAYOUT/VERIFICATION

Device layout was performed on an Apollo workstation using ChipGraph graphics editor. MOSIS two lambda (lambda = 2 microns) design rules were followed. This allowed the design rule checker on the workstation to be utilized. The layout was intended for fabrication with a self-aligned, poly-gate NMOS process using enhancement mode loads. A poly-gate PMOS process may also be used. Positive logic was implemented with 5v as logic high and 0v as logic low. The length to width ratio of all pull-up transistors was 10um/4um while all pull-down transistors were 4um/8um. All modules were full custom designs. The control unit was laid out as a programmable logic array (PLA) to facilitate quick changes as the design was modified.

Figure 1 illustrates the block diagram of the MAVERICK device which indicates the general flow of data. A complete layout of the device as completed in February 1990 is shown in Figure 2. The Appendix contains more detailed layouts of the individual modules in Figures A8 - A12

The device operation was verified by breadboarding with the 74'00 series CMOS components listed in Table 2.

Table 2

74HC00	Quad 2 Input NAND Gate
74HC02	Quad 2 Input NOR Gate
74HC04	Hex Inverter
74HC125	Quad Tri-State Buffer
74HC283	4-Bit Ripple Full-Adder
74HC373	Octal Transparent D Flip Flops
74HC86	Quad Exclusive OR Gate
74HC4040	12 Stage Ripple Counter
74HC4514	4-16 Decoder

RESULTS

Breadboard operation indicated that the original layout of the MAVERICK device would not operate correctly due to insufficient control of data with tristate buffers. This refers to the original use of tristate buffers to charge nodes to logic highs or lows. When the devices were disabled, floating nodes were created. Charge on these nodes leaked over short periods of time resulting in inconsistent device operation. Modifications to the breadboard design resolved this issue by placing registers at floating nodes to maintain logic levels. Several modifications to the original layout are therefore, necessary. This will consist of the addition of a temporary register and tristate buffer for addressable load operations. An obsolete tristate buffer will also need to be removed.

The original layout also contained some logic circuitry to test the inputs and outputs of the MAR to check when they were equal. READ and WRITE lines were enabled when this condition was true. Redesign of the device eliminated the need for this circuitry. The PLA decoder will also need to be modified to reflect the necessary changes. The original photomask layout in Figure 2 indicates the points where changes need to be made.

CONCLUSIONS

Breadboarding the MAVERICK device revealed several errors in the original photomask layout of the MAVERICK device. These errors were corrected on the breadboard to result in a fully, operational device. It is now necessary to revise the original layout to reflect the design changes. At that point, fabrication with an enhancement mode NMOS process will be possible.

ACKNOWLEDGMENTS

The author would like to thank the Microelectronic Engineering Department at the Rochester Institute of Technology for obtaining the necessary discrete CMOS parts. Thanks to Jim Kleffner of Motorola, INC in Austin, Texas for obtaining some hard to find 4-bit full adders. Special thanks to MIGUEL CHEN for his contributions to the design and layout of the MAVERICK device.

REFERENCES

[1] Kain, Richard Y.. Computer Architecture: Software and Hardware. New Jersey: Prentice Hall,1989.

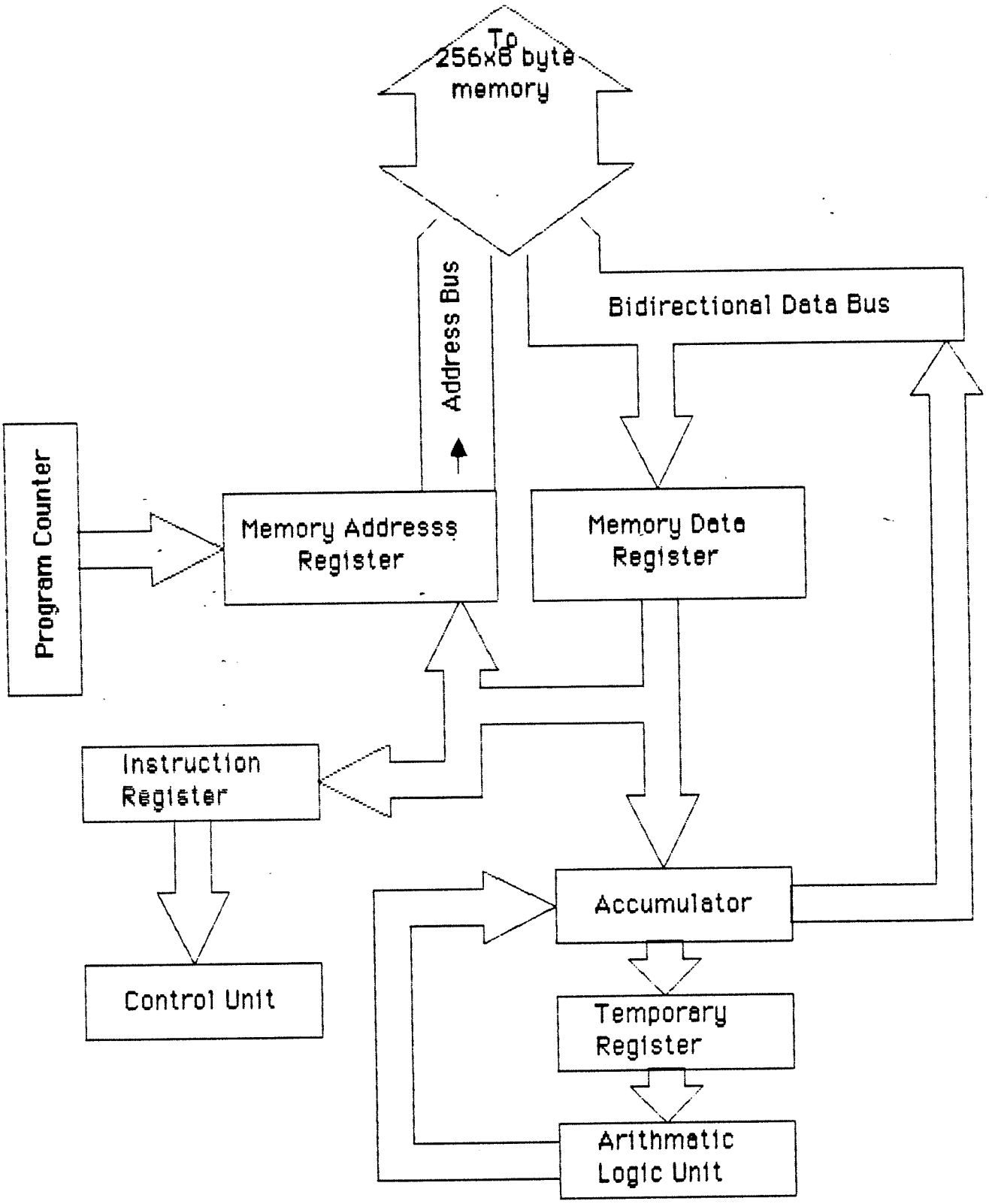

Figure 1: Block Diagram of 4-bit Microprocessor

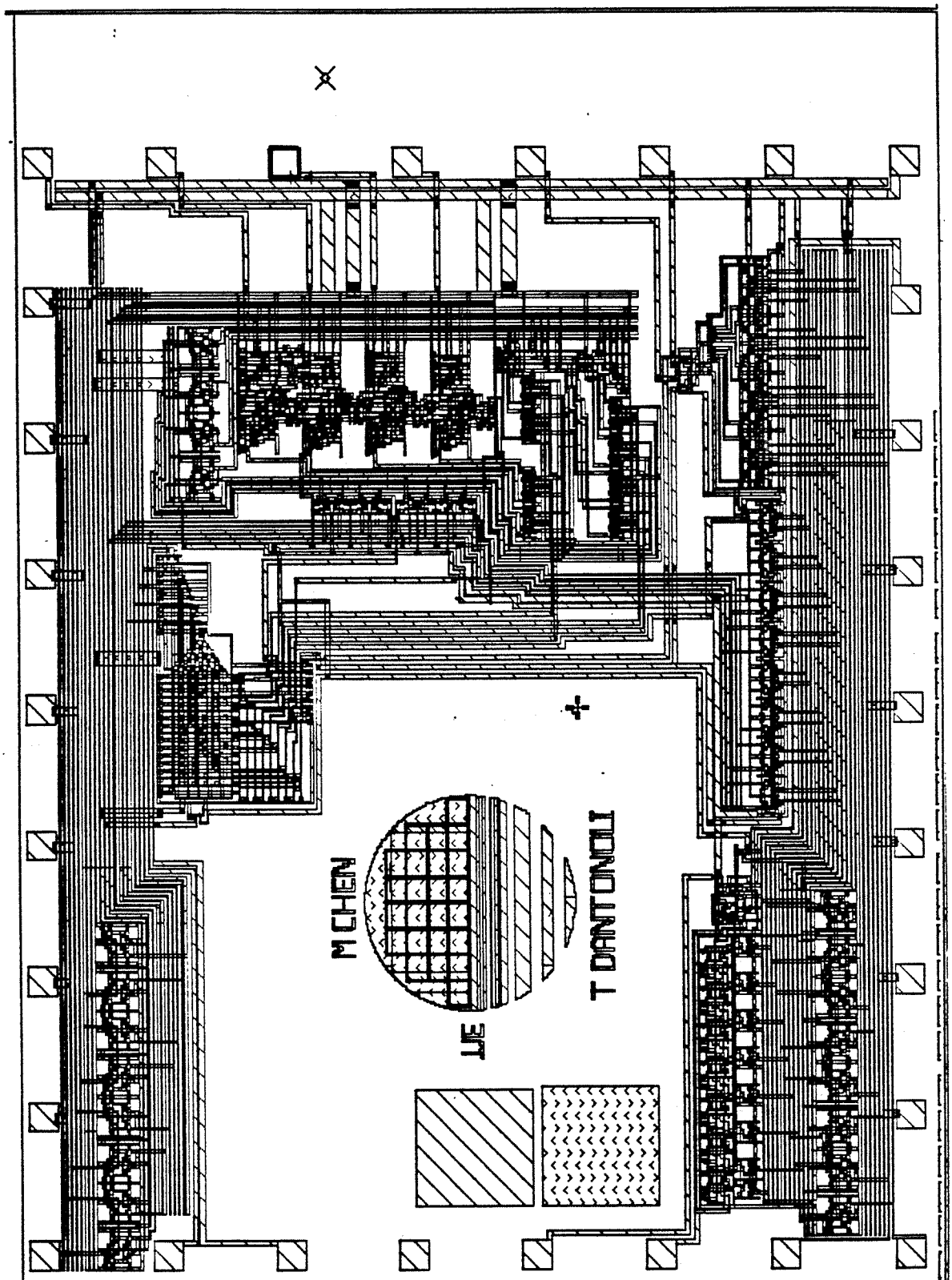

Figure 2: Layout of the MAVERICK Device