DESIGN OF A 4-BIT MICROPROCESSOR

Hsiao-Hui (Miguel) Chen
5th Year Microelectronic Engineering Student
Rochester Institute of Technology

ABSTRACT

A four bit microprocessor's layout was drawn on an Apollo workstation using MOSIS two lambda design rules. The layout's operation was verified by breadboarding with 74'00 HCMOS logic parts during Spring Quarter 1990 by Theodore D'Antonoli. Based on his findings, the layout was revised and several design changes were added. At the same time, a software simulation program (uPROSIM) for the processor was written in FORTRAN on the RIT VMS Computer System.

INTRODUCTION

Today, most of the computers are devised using digital logic rather than analog logic designs. This is mainly because the digital computers are capable of solving more problems, more easily and with greater flexibility. This is accomplished by using a simpler numerical system called the binary system. The binary number system is consisted of just 0 and 1 (logic state low and logic state high).

All computer systems can be simplified into three major components: the central processor unit (or microprocessor), input/output ports (link between processor, memory and the external world) and memory (information storage area). The microprocessor is in charge of recognizing the instructions, which are stored in the memory and retrieved through the input/output port, and executing them. The microprocessor acts as the brain in the computer system by coordinating the different components according to the instructions that it receives.

The first step of designing a microprocessor is to determine the specific functions that the device is required to perform. The individual components within the microprocessor are then designed to fulfill the functions. The typical components within the microprocessor includes program counter, memory data register, memory address register, control unit, arithmetic unit, logic unit and data buses. The interaction among them indicates the architecture of the microprocessor. The architecture used in this project is the Von Neumann architecture. This type of design is characterized by the combination of instruction and data into the same memory location (within the same data word).

Computer aided simulations are used more frequently today. Computer simulation saves time and material cost by emulating the experimental procedure in a software controlled manner without physically performing the experiment. The purpose of developing an in-house microprocessor simulation program is so that users can study and learn about the architecture of the microprocessor without having to breadboard it with discreet components. By developing an in-house simulator, it is possible to make it emulate exactly the instructions as it is performed by the microprocessor designed at RIT.

DESIGN

A 4-bit microprocessor was design in this project, the device was named as MAVERICK. The block diagrams are shown in the appendix. The instruction set is shown on table #1. <A> refers to data contained in register A and <B> refers to data contained in register B. The data codes are four bit long and so are the instruction codes (except for AST and ALD). The data and the instructions are stored in an external 256 x 8 byte DRAM. Two registers are in charge of interfacing the processor with the memory, the Memory Data Register and the Memory Address Register. Out of the sixteen possible combination from the 4-bit instruction set, two are currently unused and can be implement with additional instructions in the future.

Table #1: Instruction Code for device MAVERICK

Mnemonic	OP Code	Function	Description
DCR	0000	<A> - 1	Decrement <A> by one
INR	0001	<A> + 1	Increment <A> by one
SUB B	0010	<A> - <B>	Subtract <B> from <A>
ADD B	0011	<A> + <B>	Add <B> to <A>
LDA B	0100	<A> = <B>	Immediate load into <A>
-----	0101	----------	Unused
-----	0110	----------	Unused
HLT	0111	Halt	Stops all operation
NAD B	1000	NAND	NAND <A> and <B>
NOR B	1001	NOR	NOR <A> and <B>
INV	1010	INV	Invert <A>
NOP	1011	Wait	Increment <PC> by one
AST r	110X	Loc. r = <A>	Store <A> into location r
ALD r	111X	<A> = Loc. r	Load <a> from location r

The data, including addresses and instructions, are handled by the processor through tri-state buffers. The proper procedure to execute an instruction would be to load the instruction from memory location specified by the program counter into the control unit, enable the proper tri-states and perform the operation, and finally store the result into the accumulator (register A). The device is capable of addressing all 256 memory locations, however the last 32 memory locations' lower four bits are reserved for addressable store and load commands. As it is shown on table #1, the device is also capable of the following operations: addition, substraction, invert, NAND and NOR of four bit binary numbers. These operations can be combined (by writing a program) to perform additional tasks, such as multiplication, logic OR, exclusive OR, exclusive NOR or logic AND.

A two-phase, non-overlapping clock, clk1 and clk2, are used to time the different tri-state buffers within the processor. The voltage levels are five volts for logic state high and zero volts for logic state low. During a clk1 pulse cycle, the address from the program counter (PC) is passed onto the Memory Address bus and latched by the MAR. At the same time, the READ line of the memory device is enable and the corresponding instruction and data is loaded into the Memory Data bus and latched by the MDR. The clk2 pulse cycle decodes the instruction through the control unit and enables the corresponding logic or arithmetic device to perform the operation. It is also responsible for updating the program counter. The process is then repeated for the next clock cycle. This design permits certain degree of carelessness because the processor performs several tasks at the same time and the two clock phases do not depend on each other. In the event that the processor was unable to perform properly the operations during the clock cycle, the duration of the clock cycle can be incremented to allow the processor more time. However, during a direct write instruction, if the WRITE enable line becomes logic state high and the memory data bus or the memory address bus have not been updated to their new values then the device would write random numbers into several memory locations. In the present design, this problem is solved by adding a ring oscillator to delay the WRITE enable signal. Additional inverters can be added externally if needed.

The processor uses an eight bit word, which is composed of the instruction code and the data code. The lower 4 bits contains the data code while the higher 4 bits holds the instruction code. The only exception to the rule is the addressable instructions (store and load). These two instructions only take up the highest three bits while the other five bits hold the address in/from which the data will be stored/loaded. For example, if the instruction code 11001011 were executed then the processor would store the content of register A into memory location 11101011. The first three bits of the word contains the instruction and the last five bits contains the lower five bits of the address. The upper three bits of the address will always be 111 for the addressable load instruction to indicate its

location in upper memory. The same argument holds true for the addressable load instruction. Since the data is limited to four bits in length, the highest number that can be stored or loaded in any memory location would be 15 (2^4, sixteen numbers, from 0 to 15).

Tri-state buffers were used to allow the capability of having several devices sharing the same data bus. Tri-state buffer functions as isolation between the device and the data bus. When the tri-state buffer is disable, the output is at a high impedance state which causes the output terminal to behave as a virtual open circuit. The tri-state buffers' enable line are governed by the control unit, which dictates which device has access to the data bus.

Other components designed for the MAVERICK device include a program counter, shift registers, an arithmetic unit, a logic unit (NAND, NOR, INV) and a control unit. The circuit layout of these devices are shown in the appendix section. The program counter is based on eight D flip-flops. The circuit drawing and the layout of the counter are shown at the end of this report (figure 3 and 4). The clock signal is fed into the first D flip-flop and the rest of the flip-flops are connected in a cascaded form with the clocks connected to the output Q signal of the previous stages. All the D input terminal are connected to the output Q bar signal, the Q bar signal acts as the feedback to oscillate the D input signal between logic state high and low. The output is taken from the Q bar signals.

The shift register are also based on D type flip-flops except that they lack of PRESET and CLEAR terminals. When the register is enable, the input signal is passed onto the output, and when the register is disable, it latches the input. The arithmetic logic unit (ALU) is capable of handling four bit numbers. The logic module perform a bit by bit logic operation on the contents in the accumulator. The arithmetic module is composed of a standard four bit ripple full adder. The control unit was designed using a PLA design architecture. The instruction signals (first four bits of the Memory Data Bus) and the clock 2 signal (clk2) are connected to the input of the PLA as the select lines. The output of the PLA are connected to the enable terminals of the shift registers and the tri-state buffers. Logic state high enables these devices and logic state low disables them (positive logic).

LAYOUT/LAYOUT REVISION

The original layout design was drawn during the Fall and Winter Quarters 1989-1990. The layout was performed on the Apollo workstation using Chipgraph graphics editor. MOSIS two Lambda (Lambda = two microns) design rules were followed. DRACULA design rule checker was used for each of the individual device layouts, however the entire final layout was never electronically check for

design rule errors due to the enormous size of the file. The layout was intended for fabrication with a self-aligned, poly-gate NMOS process using enhancement mode loads. The length to width ratio of all pull-up transistors are 10um/4um while all pull-down transistors are 4um/8um. The entire layout was a full custom design from transistor level up.

During Spring Quarter 1989-1990, Ted D'Antonoli breadboarded the device and found several logic errors in the original design. These errors were corrected during Summer Quarter 1989-1990. The main corrections included the redesign of the program counter, re-mapping of the layout and addition of a temporary register. The re-mapping of the layout will allow a wider degree of freedom in terms of future addition or partial redesign of single components. The new layout will be able to accommodate newly designed devices to be placed into the processor without having to change the rest of the layout.

MICROPROCESSOR SIMULATION SOFTWARE

uPROSIM (nickname GOOSE) is a software simulation program written in FORTRAN on the RIT VMS computer system. The objective of this simulation program is to allow users to simulate the device MAVERICK on a computer system rather than breadboarding the processor with discreet components. The advantages to use a simulation program for this type of experiment are: software simulation will decrease the experimental setup time considerably and it also makes the need for discreet components obsolete.

A complete copy of the user manual and a printout of uPROSIM can be found at the end of this report. The printout corresponds to the Alpha version of the program, which is currently functional (August 8, 1990). It is capable of reading assembly codes by either user interactive entry mode (keyboard) or file entry mode. The output of the program reports the address code in both binary and decimal numbers. The instruction codes are in the forms of binary numbers and mnemonic codes. The program can be easily modified to suit future additions to the MAVERICK microprocessor or any other microprocessor that might be designed at RIT. The flowchart of the program can be found at the end of this report as well. The structure and the logic of the program is identical to the logic design behind the MAVERICK device, by doing so, users can test and experiment with the program as it were the processor itself without having to deal with complicated and confusing hardware setups. Future additions to the program are intended, the next release will be able to read in assembly codes in binary number, mnemonics or hexadecimal numbers.

At the present time, the program and a demo data file can be obtained through the VAX system by coping the following files: [HNC1389.UPROC]uPROSIM.EXE and [HNC1389.UPROC]DEMO.DAT. These two files should be used in conjunction with the user manual.

RESULTS

The design errors pointed out by Ted D'Antonoli in his paper [1] were resolved by re-designing few components and revising the original MAVERICK device layout. The new layout (VIPER) includes a new eight bit program counter with PRESET and CLEAR lines, a new Programmable Logic Array as the control unit and few registers and tri-state buffers were also added. The new layout possesses all the characteristics of the original layout, the same design rules and the same fabrication process are still valid.

A software emulator was written for the MAVERICK microprocessor. The existence of this software will enable the possibility of quick and easy testing of device's logic for those who desires to improve the design. The new program counter was design from transistor level up using the QUICKSIM simulation software on an Apollo workstation. Expanded simulation files exist as reference for future work. The time constraint did not allow the entire circuit to be simulated, however the existing program counter's files can serve as templates to setup simulation for the rest of the MAVERICK device.

CONCLUSION

A four bit microprocessor was designed, breadboarded, simulated, emulated and drawn during the course of school year 1990. The breadboarding and the simulations provide sufficient data to prove that the design is operational as intended. The microprocessor MAVERICK's present characteristics reach beyond its original specifications both in operations and the size of addressable memory locations. MAVERICK is the first of its kind at RIT and it was created to be the milestone for other microprocessor design that might follow. When MAVERICK and uPROSIM are used as templates for other designs, the overall time required to generate other distinct design can be reduced to a single academic quarter. Immediate future related projects that should be taken into consideration include fabrication and testing of the device.

ACKNOWLEDGMENT

I would like to thanks the Microelectronic Engineering Department for this wonderful opportunity and for its support to the project and the Computer Engineering Department for allowing the access to the Apollo Workstation and for the lab technicians for their help. The original idea of the project came form Dr. Lynn Fuller, who also served as advisor for some period. Rob Pearson assisted and advised on redesign of the program counter and came out with the concept of uPROSIM. Special thanks goes to Ted D'Antonoli for his contribution to the design and the endless hours spent in the Apollo workstation room during the busiest time of his life.