Design of a Low Power 80386 Based System

by

Craig A. Robson

BSEE, Kansas State University, 1987

A MASTER'S THESIS

submitted in partial fulfillment
of the requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1989

Approved by:

Donald A Gaskey

Major Professor

Contents

Figures

Tables

Acknowledgments

This research was accomplished through a contract sponsored by Sandia National Laboratories, Albuquerque, New Mexico, with the support of Kansas State University.

I would like to thank my major professor, Dr. Donald Lenhert, for his guidance and advice on this and other research projects throughout my education at Kansas State University. Additionally, I would like to thank all of the Electrical Engineering Faculty for their challenge and instruction over the past several years.

I would also like to thank my fellow graduate students whose encouragement and suggestions have helped me to complete my education. Especially I want to thank the graduate students who I have shared an office with over the past two years.

Finally, I wish to thank my parents, David and Jolene, for their encouragement and support throughout my entire education.

1 Introduction

This thesis outlines the design of a high-speed, low-power microprocessor based system. This system is designed to be used in a satellite instrumentation and data acquisition system which will have several similar systems sharing a common bus. Most of this thesis details the design of a memory module which is designed to work with the 80386 and other 32-bit microprocessors. The rest of the thesis shows the interface of this memory module to an existing 80386 system. This thesis outlines only the hardware design, from which an actual circuit and schematics can be derived.

2 Design Constraints

My design must meet several constraints. The first is that it must be based on Intel's M80386 high speed microprocessor. This is the military version of Intel's popular commercial 80386 microprocessor. The 80386 is a 32-bit CMOS microprocessor designed to be downward compatible with Intel's iAPX86 family of microprocessors. There is a large base of microcomputers designed around the older iAPX86 microprocessors which means there is already a vast base of software and hardware developed around this chip. The M80386 is detailed in several Intel publications including the M80386 data sheet and the 80386 Hardware Reference.[1,2]

In the mid-1980s, IBM began marketing a microcomputer based on Intel's 80286 microprocessor. These microcomputers are called AT's and have a 16-bit bus for peripheral data flow. This 16-bit bus operates at 8 MHz. My design is to be based on a modified AT bus. The modified AT bus is essentially the standard AT bus with additional signal lines to allow multiple processors to use the same AT bus.

Chips and Technology, Inc. makes a five chip set that allows easy implementation of the AT bus with the 80386 microprocessor. This chip set controls the 8 MHz AT bus and coordinates communication between it and a faster 16 MHz local microprocessor bus. My design is to be based on this Chips and Technology chip set.

Another design constraint is that the system is to be designed using radiation hardened CMOS parts wherever possible. This will help reduce power consumption and will decrease the effects of radiation that might be encountered in a satellite environment.

To help reduce power consumption, the system is to run at a low clock speed when idle. The system will switch to a higher processor speed when computational speed is needed.

Since this system is to be combined with several other similar systems, a common 32-bit memory module is to be designed for usage in all systems. The other proposed systems are currently based on AT&T's DSP32C digital signal processing chip and National Semiconductor's 32532 Advanced 32-Bit Microprocessor. This common memory module is to use high-speed, low-power SRAM, and must incorporate

a modified Hamming code error detection and correction unit.

The following table restates the initial design constraints placed on my design.

Table 1: System Design Constraints

- M80386 based.
- Based on a modified AT bus.
- Use a Chips and Technology Chip Set.
- Use Radiation Hardened CMOS, when possible.
- Must have Low Power Consumption.
- Clock Speed Must Be Switchable.
- Must Use a Common Memory Module with Error Correction.

3 Common Memory Module

A common memory module is to be used with three different microprocessors in the system. All three are high-speed CMOS, 32-bit microprocessors with similar memory interfaces. The memory module is designed to easily interface to all three microprocessors.

3.1 Memory Module Specifications.

The memory module is to contain 512k 32-bit words of static random access memory. The memory module is to include a modified Hamming Code error detection and correction unit. The modified Hamming coding scheme will require seven extra bits to encode the data in memory. To store these bits, the memory module actually has 512k 40-bit words, with one bit unused in each word.

3.2 Memory Module Interface.

The interface signals for the common memory unit are outlined in Table 2. The module is designed to act like an asynchronous memory, under normal conditions. The ALE signal and the BDSEL signal indicate the initiation of a memory cycle for the module. The ALE signal must not be asserted until all address, data, and control lines are available. The module will generate the WAIT signal to extend the duration of a memory cycle when necessary. This WAIT signal will only occur when a memory error occurs or when a byte-write operation is requested.

Table 2: Memory Module Interface Signals

Signal	Type	Description
ADDR18-2	Input	Address Bus. 17 bit address bus for 1 of 512k word selection.
ALE	Input	Address Latch Enable. A high value indicates that the address is valid.
BDSEL	Input	Board Select. Selects the memory module as the current module when high.
BE3'-0'	Input	Byte Enables. A low input on one of these lines indicates a memory access to the corresponding byte. These signals are used only during write operations.
Clk	Input	Clock. Clock for the memory module state controller. For the 80386, this is an 8 MHz clock.
DATA31-0	I/O	Data Bus. 32-bit bidirectional data bus for the module.
ERROR'	Output	Error. A single-bit correctable error occurred and was corrected.
GND		Ground. Ground for the memory module. This will require multiple pins on the module depending on the module's total power requirements.
MERROR'	Output	Multiple Error. A multi-bit error was detected. The data returned is erroneous.
R/W'	Input	Read/Write Signal. When high the requested access is a read. When low, a write is requested.

Vcc		Power. Five volt power supply for the module. This will require multiple pins depending on the total power requirements for the module.
WAIT	Output	Wait. When this signal is high the board is requesting an extension of the normal cycle time. This signal is normally low.

****

3.3 Memory Module Design

Figure 1 shows the memory module design. The interface on the left is an edge connector for interfacing the module to the microprocessor systems. The EDC, IDT49C460, is Integrated Device Technology, Incorporated's 32-bit CMOS error detection and correction unit. It generates seven check-bits during a write operation and checks for errors and corrects them during read operations. The coding scheme used is detailed in the EDC Data Sheet.[3] For maximum fault tolerance of the memory, the seven check bits generated by the EDC can be spread out among the forty bits in each word.

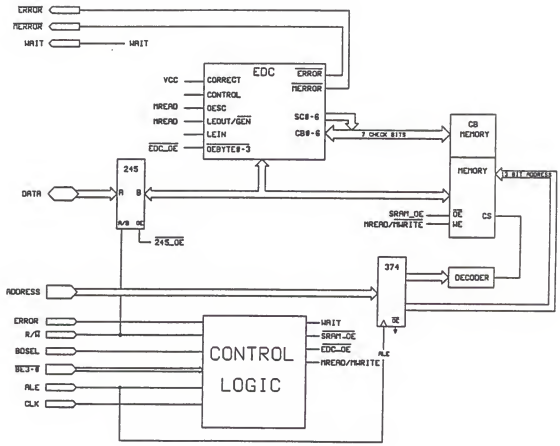

Figure 1: Memory Module

The ALE signal indicates the start of a memory cycle, at which time the address bus is latched into a register. The decoder then generates the chip select signal for the memory, and controller determines the type of operation. If the operation is for the current memory module, it generates the needed control signals. In the read case, the EDC will verify that the data is correct, and in the write case, the EDC will generate check bits. The controller will generate the WAIT signal as necessary.

3.4 Memory Module Control.

The memory module control logic controls the EDC unit, SRAM, latches, WAIT signal, and buffers. During normal operation, 32-bit accesses with no errors detected, the controller does very little. However, in the event of a byte write operation or if a read error is detected, the control unit schedules a module bus switch operation and controls WAIT signal generation.

All read operations are 32-bit operations, all four bytes of the 32-bit word will be output from the module. It is the microprocessor interface's responsibility to latch the byte or bytes desired. If a single-bit error occurs, the memory module corrects the data that is output from the module, asserts ERROR', and writes the corrected

data back to memory. If a multiple-bit error occurs, the module outputs erroneous data and asserts the MERROR' signal.

The module is designed to only use the clock to control the module when an error occurs or when a byte-write operation is performed. In these cases, the module will use the clock to control the generation of the WAIT signal and to control the generation of new valid data in the static memory devices. This gives the maximum flexibility for use with other microprocessors running at different clock rates.

There are four types of memory operations. They are:

- **32-bit write operation.** For this operation, the EDC generates seven check bits and a new 40-bit word is written to SRAM.

- **32-bit read operation with no errors detected.** For this operation, a 40-bit read is performed and the EDC verifies that the 32 data bits are correct.

- **32-bit read operation with an error detected.** In this case, a 40-bit read is performed and the EDC detects a single-bit error. The EDC will correct the bit in error and the corrected data is

written back to memory and is output from the module. The module must assert the WAIT signal while this extra operation is taking place. To simplify control sequencing of this operation, this operation also occurs when a multiple-bit error is detected.

- Byte write operations. This case poses an extra problem because the EDC uses a modified Hamming code which creates check-bits based on a full 32-bit word. Since the microprocessor is supplying only 8, 16, or 24 bits, the remaining byte or bytes must be obtained from memory. So, the module must first read the current contents of memory and correct any single bit error encountered. The EDC will latch this corrected 32-bit word into its output latch. Finally, the new bytes from the microprocessor and the other old bytes, possibly corrected, from the EDC are placed on the memory module's data bus. The data and new check bits which reflect the contents of the module's internal data bus are written to memory.

The 32-bit read and write operations are handled without clock sequencing. The 32-bit read with error and byte-write operations require sequencing by the clock. Figure 2 is the state diagram for the memory module controller. The diagram is broken into three different cases:

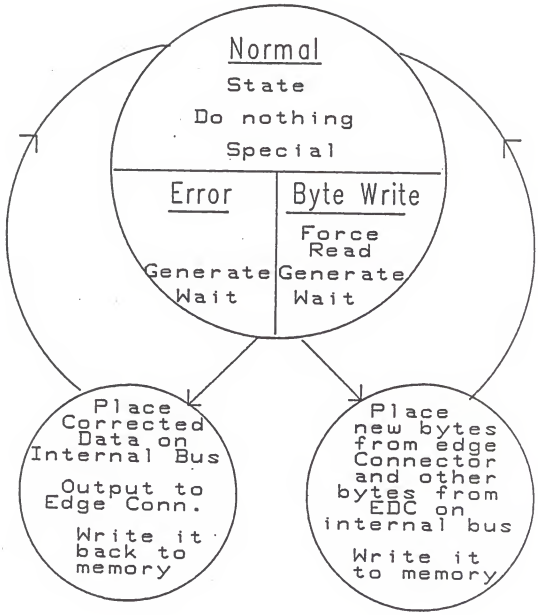

Figure 2: State Diagram for the Memory Controller.

1. The first case is the normal case. In this case, the memory controller is performing one of five distinct things. It is either doing nothing, performing a 32-bit read, performing a 32-bit write, performing the read part of a 32-bit read with an error, or it is forcing a read as the first part of a byte write operation.

2. The second case is the forcing of a write following a read with error. This is necessary to correct the data in memory. To make the control logic simpler, this operation is also performed when a multiple-bit error is detected.

3. The last case is the write portion of a byte-write operation.

Note that although the state diagram is broken into three distinct states, the lower two states are actually performing the same write operation. This allows the controller to use a simple flip-flop to control the sequencing. The normal case is identified as STATE_0 and the forced write case as STATE_1. The lower two states can be differentiated by checking for a byte-write operation.

The memory module control logic controls the module's bus sequence for byte-write operations and error correction operations. The controller generates the following signals.

- **MREAD/MWRITE'** This signal is the WE' signal for the memory and also controls the EDC's check-bit output and generation. It is basically a read/write' signal for the SRAM.
- **245_OE0-3'** These are the output enable signals for the module's I/O buffers. These are active-low signals.
- **EDC_OE0-3'** These are the EDC's data output enable signals. These are active-low signals.
- **SRAM_OE'** This is the output enable signal for the SRAM. This is an active-low signal.
- **WAIT** This is the WAIT signal to the edge connector. The WAIT signal timing is such that it can be used to generate the READY# signal for the 80386 microprocessor.

The WAIT signal is asserted when there is an error detected or when a byte write operation is requested. It is

WAIT = BDSEL AND

((byte_write AND STATE_0) OR (ERROR')')

The MREAD/MWRITE' signal controls the SRAM and EDC as shown in Figure 1. It is

MREAD/MWRITE' = (byte_write AND STATE_0) OR
(R/W' AND byte_write' and STATE_0).

Where byte_write is an internal signal added for readability. It is

byte_write = (R/W')' AND (BE0' OR BE1' OR BE2' OR BE3')

and can be combined in the equation for each signal requiring byte_write.

The SRAM_OE' signal is the output enable signal for the SRAM. It is the inverse of MREAD/MWRITE

SRAM_OE' = ((byte_write AND STATE_0) OR
(R/W' AND byte_write' and STATE_0))'.

The EDC_OEn' signals are the EDC's OE' signals for a specified byte. The EDC's contents are used only for a byte-write operation or for error correction. EDC_OEn' is

```
EDC_OEn' = ( (STATE_1 AND byte_write')  OR
             (STATE_1 AND byte_write AND (BEn')' )'.
```

The 245_OEn' signals are the I/O buffers' output enable signals. They are

```
245_OEn' = ( BDSEL AND (
           (byte_write' AND STATE_0)  OR
           (byte_write  AND STATE_1 AND BEn') ) )'.
```

The D flip-flop value is controlled by the following equation. It is loaded each rising edge of the CLK signal. The controller only enters STATE_1 for one cycle at a time.

```
STATE_1/STATE_0'.d = (ERROR')' OR
                     (byte_write AND STATE_0).
```

These control equations can be implemented with two programmable logic devices. Appendix A contains source files for one possible implementation.

3.5 Devices Used in the Memory Module.

The components on the memory module are all CMOS devices. The following devices, used for timing and power calculations, are recommended for the module. All devices except the PALs are made by Integrated Device Technology, Inc. The PALs for the controller and address decoder are manufactured by Cypress Semiconductor, Inc.

- 245. IDT54FCT245A. Fast CMOS Non-Inverting Buffer Transceiver. Four of these devices are used to buffer the data bus in and out of the memory module.[4]

- 374. IDT54FCT374A. Fast CMOS Octal D Register (3-State). Three of these devices are used for the address latch.[5]

- Control. PALC16R4, PAL16L8. These programmable devices can be used to generate the control logic for the module. These devices should only be used for module prototypes and can be replaced by discrete logic in a final version of the memory module. One of each is required for the memory module controller.[6]

- Decoder. PALC16L8. This programmable device can be used to generate the address decoding for the module. This device should only be used for module prototypes and can be replaced by discrete logic in a final version of the memory module.[7]

- EDC. IDT49C460B. 32-bit CMOS Error Detection and Correction Unit. This device handles the Modified Hamming Code generation, error detection and correction.[8]

- SRAM. IDT7164L35B. Thirty-five nanosecond CMOS Static RAM (8K x 8-bit). Forty of these will be used in the module prototype. These memories are fast, but a faster version can be used in some implementations for improved performance depending on the clock speed for the module.[9] A final version of the module with 512K 40-bit words will require 320 of these 8K x 8-bit chips or 80 32K x 8-bit SRAMs if available. The number of SRAM chips should not have a large effect on the module's power requirements because only five SRAMs will be active at any time. The standby power requirements for the remaining SRAM chips are negligible.

3.6 Power Requirements.

The components on the memory module are all CMOS devices. Power requirements, for the module built using the components outlined above, are shown below. These requirements are calculated according to Integrated Device Technology's method which does not consider the effects of capacitive output-loading.

The IDT54FCT245A's power requirements are specified by the power supply current formula on page 10-84 of the data sheet.[10] It is

$$I_C = I_{QUIESCENT} + I_{INPUTS} + I_{DYNAMIC}$$

where

$$I_{QUIESCENT} = I_{CC},$$

$$I_{INPUTS} = 0,$$

and

$$I_{DYNAMIC} = I_{CCD} (f_i N_i)$$

where

I_{CC} is the quiescent power supply current,

I_{CCD} is the dynamic power supply current,

f_i is the input frequency,

and

N_i is the number is inputs changing at f_i.

All currents are in milliamps and all frequencies are in megahertz.

Since my design uses only CMOS parts, I_{INPUTS} is zero because no current must be sourced for other TTL devices. Assuming that all eight inputs change at 8 MHz, the fastest memory operation frequency for the 80386, the following current and power are required for each 245.

$$I_C = 2.0 + 0.25(8)(8)$$

$$= 18 \text{ mA.}$$

$$P = I_C V_{CC}$$

$$= 18 \times 5$$

$$= 90 \text{ mW.}$$

The four 245s required by the module will require 360 mW.

The IDT54FCT374's power requirements are calculated similarly to the 245s. But, IDT includes an extra term, $f_{CP}/2$, in their dynamic power supply current formula. This term accounts for register clocking in the device, where f_{CP} is the clock frequency. This formula is detailed in the 374 data sheet on page 10-101[11]. The supply current is

$$I_C = I_{QUIESCENT} + I_{INPUTS} + I_{DYNAMIC}$$

where

$$I_{QUIESCENT} = I_{CC},$$

$$I_{INPUTS} = 0,$$

and

$$I_{DYNAMIC} = I_{CCD} (f_{CP}/2 + f_i N_i).$$

Assuming once again that all eight inputs can have transitions at 8 MHz, and that all eight registers, which are clocked by ALE, are clocked at 8 MHz, the current and power required are

$$I_C = 1.5 + 0.25[(8/2) + 8(8)]$$

$$= 18.5 \text{ mA}$$

and

$$P = I_C V_{CC}$$

$$= 18.5 \times 5$$

$$= 92.5 \text{ mW}.$$

The three 347s require 277.5 mW.

The controller PALs require a supply current governed by the equation on page 4-10 of the PAL C 20 Series data sheet[12]. The dynamic current is

$$I_{CC(AC)} = I_{CC(DC)} + (0.6)(f_{OP})$$

where

$I_{CC(DC)}$ is the quiescent operating current

and

f_{OP} is the operating frequency.

Using an 8 MHz clock frequency, the following current and power are required for each PAL.

$$I_{CC(AC)} = 70 + (0.6)(8)$$

$$= 74.8 \text{ mA}$$

$$P = I_{CC(AC)} V_{CC}$$

$= 74.8 \times 5$

$= 374$ mW

The two controller PALs will require 748 mW.

The decoder PAL requires a supply current calculated the same as the controller PAL.

$I_{CC(AC)} = 70 + (0.6)(8)$

$= 74.8$ mA

$P = I_{CC(AC)} V_{CC}$

$= 74.8 \times 5$

$= 374$ mW

The Error Detection and Correction Unit's DC characteristics are outlined on page 8-248 of the EDC's data sheet[13]. Because we are using only CMOS devices, the power required by the EDC will be

$$I_{CC} = I_{CCQ} + I_{CCD}(f_{OP})$$

where

I_{CCQ} is the quiescent power supply current,

I_{CCD} is dynamic power supply current,

and

f_{OP} is the operating frequency, in Megahertz.

The EDC may operate at 8 MHz. Under this assumption, the current and power required are

$$I_{CC} = 5 + 10(8)$$

$$= 85 \text{ mA}$$

and

$$P = I_{CC}V_{CC}$$

$$= 85 \times 5$$

$$= 425 \text{ mW}.$$

The forty SRAM devices are divided into 5 active devices and 35 devices on standby. The current and power for an active 35 ns military part, from page 4-96 of the data sheet[14], are

$$I_{CC} = 140 \text{ mA}$$

and

$$P = I_{CC}V_{CC}$$

$$= 140 \times 5$$

= 700 mW.

The SRAM that is on standby requires 0.005 mA and 0.025 mW to retain its data so these devices can be excluded from power estimates. With this assumption, the five active SRAMs will require 2.1 watts.

The total power requirement for the module prototype is outlined in Table 3. A final memory module which contains 32K x 8-bit SRAMs which require 1 watt will use the power outlined in Table 4.

Table 3: Prototype Memory Module Power Requirements.

Device	Number Used	Total Power Required
245	4	360 mW
374	2	278 mW
Control	2	748 mW
Decoder	1	374 mW
EDC	1	425 mW
SRAM (active)	5	2100 mW
SRAM (standby)	35	0.001 mW
Total Power for Module		4.28 W

Table 4: Final Memory Module Power Requirements.

Device	Number Used	Total Power Required
245	4	360 mW
374	2	278 mW
Control	1	374 mW
Decoder	1	374 mW
EDC	1	425 mW
SRAM (active)	5	5000 mW
SRAM (standby)	35	ignored
Total Power for Module		7.28 W

The power estimates for the modules are worst case values that do not include output loading. These values are higher than desired, but they are only theoretical maximums and the modules may require less at any given time.

3.7 Memory Module Timing.

Figures 3, 4, and 5 are memory module timing diagrams. Several basic assumptions were made in the diagrams.

- The address, and the R/W' and the BE signals are all valid when ALE is asserted.

- The module requires one 8 MHz clock cycle for normal operation and two 8 MHz clock cycles for error correction and byte write operations.
- The WAIT signal meets the setup times for the controller so that the controller can sequence the EDC operations using a 8 MHz CLK.
- The data from the module is latched on the rising edge of the CLK signal.

3.7.1 Module Read Timing.

Figure 3 shows the memory module timing for a normal 32-bit read cycle. The cycle starts when the ALE signal is asserted. This latches the address into the 374's and gives a point of reference for control signal timing.

Interface timing parameters for the module are shown below. There are six different signal paths to consider for read timing. They are calculated below.

One critical assumption has been made in the timing equation specifications. It is that the read operation occurs fast enough for the module's clock to control the sequencing of the error correction. This assumption will be valid if the module timing is such that the

microprocessor is able to use the module in zero wait state operation. If this is not true, the module should divide the clock in two or use some other method to delay checking for the ERROR' signal internally.

The module parameters are calculated for the devices listed in Section 3.5 and under the assumption that all combinational logic is generated by 25 ns PALs and that the chip select logic is also generated by a 25 ns PAL.

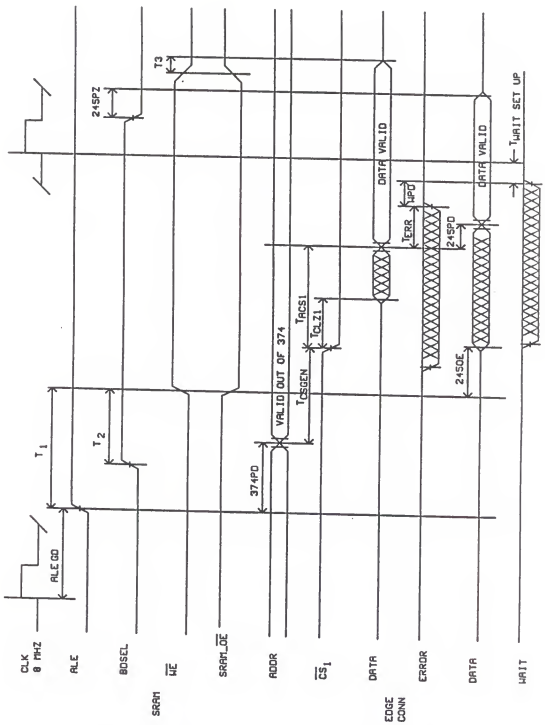

Figure 3: Memory Module Read Cycle Timing

Four of the timing paths relate to the ALE signal. Two different conditions must be satisfied before DATA becomes valid to the module's edge connector, and two different conditions must be satisfied before the WAIT signal is valid at the edge connector. For each of these signals, both conditions can be calculated and the worst case condition can be used for the module's timing specification.

The ALE to DATA valid delay depends on both the SRAM output enabling and the SRAM decode logic. These two conditions are

$$\begin{aligned} \text{ALE2DATA} &= T_1 + T_{OE} + 245_{PD} \\ &= 25 + 20 + 4.9 \\ &= 49.9 \text{ ns} \end{aligned}$$

and

$$\begin{aligned} \text{ALE2DATA} &= 374_{PD} + T_{CSGEN} + T_{ACS1} + 245_{PD}. \\ &= 7.2 + 20 + 35 + 4.9 \\ &= 67.1 \text{ ns}. \end{aligned}$$

So ALE2DATA is 67.1 ns, which is the worst of the two cases.

The ALE to WAIT valid delay also depends on both the SRAM output enabling and SRAM decode logic. It is basically a delayed version of ALE2DATA. The conditions for ALE2WAIT are

$$\text{ALE2WAIT} = T_1 + T_{OE} + T_{ERR} + T_{WAITPD}$$

$$= 25 + 20 + 28 + 25$$

$$= 98 \text{ ns}$$

and

$$\text{ALE2WAIT} = 374_{PD} + T_{CSGEN} + T_{ACS1} + T_{ERR} + T_{WAITPD}$$

$$= 7.2 + 20 + 35 + 28 + 25$$

$$= 115.2 \text{ ns.}$$

So ALE2WAIT is 115.2 ns, which is the worst of the two cases.

The DATA and WAIT signals both depend on the BDSEL signal. The BDSEL signal is primarily an output enable signal, but it also controls the SRAM outputs. The two output dependencies are

$$\text{BDSEL2DATA} = T_2 + T_{OE} + 245_{PD}$$

$$= 25 + 20 + 4.9$$

$= 49.9$ ns

and

$$BDSEL2WAIT = T_2 + T_{OE} + T_{ERR} + T_{WAITPD}$$

$= 25 + 20 + 28 + 25$

$= 98$ ns.

The DATA from the memory will be valid until the microprocessor request ends the module's cycle. At this time the BDSEL signal should be unasserted. The data is valid until it enters a high-impedance state or until the SRAM is deselected. At which time the 245 buffer will provide garbage data. The following two parameters are important. The control signals are the signals from the 80386.

$$BDSEL2HIGHZ = T_3 + 245_{PZ}$$

$= 25 + 1.5$

$= 26.5$ ns minimum

$= 25 + 6.0$

$= 31.0$ ns maximum

$$CONTROL2NODATA = T_4 + T_{OHZ} + 245_{PD}$$

$$= 25 + 0 + 1.5$$

$$= 26.5 \text{ ns}$$

Table 5 summarizes the memory module's read characteristics. It is apparent that the generation of the WAIT signal has the longest delay. This is important because the microprocessor can not use the data until it is validated by WAIT. The long time is attributed to the EDC which must test the data for errors.

Table 5: Memory Module Read Timing Parameters

Parameter	Time
ALE2DATA	67.1 ns. max.
ALE2WAIT	115.2 ns. max.
BDSEL2DATA	49.9 ns max.
BDSEL2WAIT	98.0 ns max.
BDSEL2HIGHZ	26.5 ns min. 31.0 ns max.
CONTROL2NODATA	26.5 ns min.

3.7.2 Module Write Timing.

Figure 4 shows the memory module's timing for a normal 32-bit write cycle. The cycle is initiated when the ALE signal is asserted. This latches the address into the 374's and guaranties that all control inputs are correct. Since the EDC must generate check-bits which are part of the data, the data is not valid to the memory until the check-bits are valid.

Interface timing parameters for the module are shown below. There are several different signal paths to consider for the write timing.

The module parameters are calculated using the devices listed in Section 3.5 and under the assumption that all combinational logic is generated by 25 ns PALs and that the chip select logic is also a 25 ns PAL.

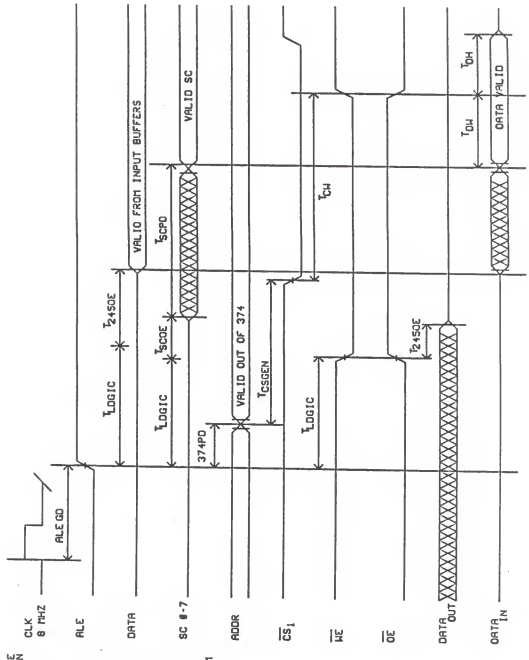

Figure 4: Memory Module Write Cycle Timing

The critical parameters are the time from ALE to data being latched. This length of this time is dependent on two equations.

$$ALE2LATCHED = T_1 + 245_{OE} + T_{SCPD} + T_{DW}$$

$$= 25 + 6.5 + 28 + 15$$

$$= 74.5 \text{ ns minimum}$$

or

$$ALE2LATCHED = T_2 + T_{SCOE} + T_{SCPD} + T_{DW}$$

$$= 25 + 27 + 28 + 15$$

$$= 95 \text{ ns minimum}$$

So, the minimum write cycle time from ALE to data latched is 95 ns.

3.7.3 Module Error Cycle Timing

Figure 5 shows the module's timing for an error cycle. The timing for STATE_0 is similar to the above read timing and will not be validated in this thesis. The timing for STATE_1 is similar to the above write timing, but it must occur in one 8 MHz clock cycle.

The critical time is the writing of corrected data to memory

$$1 / f_{CLK} > T_3 + EDC_{OE\_byte} + EDC_{SCPD} + T_{DW}$$

$$125 > 25 + 25 + 28 + 15$$

$$125 \text{ ns} > 93 \text{ ns}.$$

This shows that the module can write the data back to memory in the allotted amount of time.

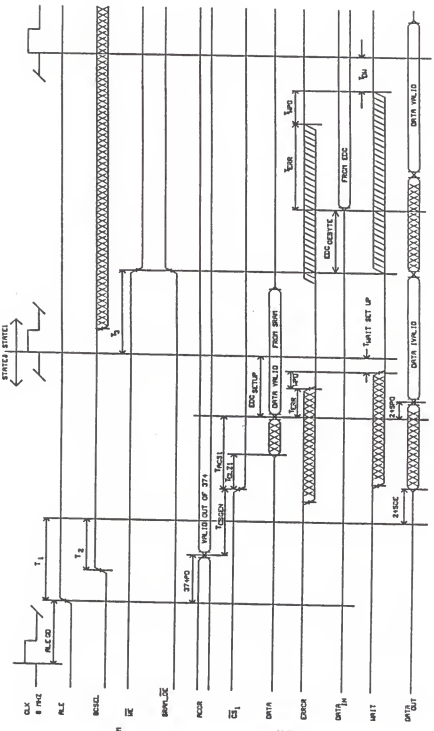

Figure 5: Memory Module Error Cycle Timing

4 CS8232-16 CHIPSet

Many existing microcomputers are based on an AT bus configuration. IBM originally designed this configuration around Intel's 80286 microprocessor. This configuration is based on a 16-bit system bus with a clock speed of 8 MHz. Chips and Technologies, Incorporated makes a set of seven VLSI chips which will handle most of the control logic for an 80386 based AT type microcomputer. This chip set is designated as the CS8382: AT/386 CHIPSet. A large portion of this chip set and one other chip, Chips and Technology's 82C206, are used to control arbitration and data communication between the AT 8 MHz bus and the 80386's 16 MHz local bus.[15,16]

4.1 Changes to the CS8232-16 CHIPSet

Figure 6 shows the basic Chips and Technology AT/386 system. There are four data buses. The local data bus is a 32-bit, 16 MHz bus which interfaces the 80386. Memory is attached to the memory data bus which is also a 32-bit, 16 MHz bus. The I/O channel and peripheral data buses are 16-bit buses operated at 8 MHz. The I/O channel and peripheral data buses are multiplexed to use the same wires. A major role of the Chips and Technology chips is

to control data flow between these high and low frequency buses. The other major function that these chips are designed to handle is a dynamic memory, DRAM, interface.

Since the Chips and Technology chips are designed to use dynamic memory for the main system memory, a modification is needed to control the static memory. A large effort was required to make this change. It basically consisted of removing the 82C302 Chips and Technology dynamic memory control chip and designing a replacement.

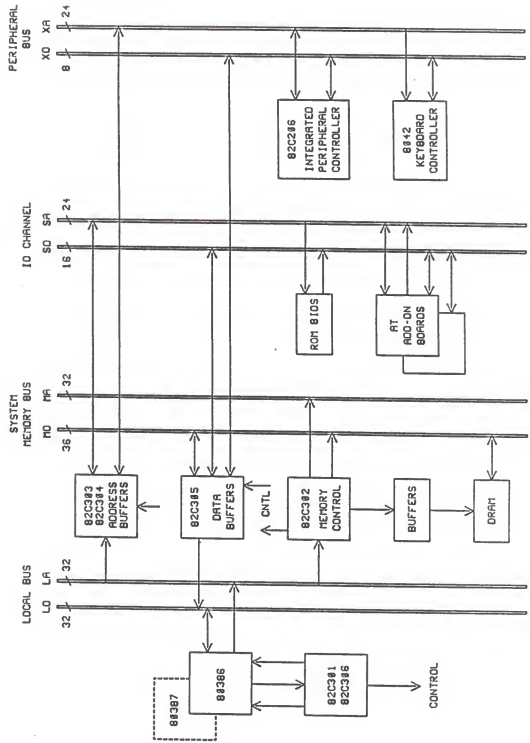

Figure 6: Original Chips and Technology Design

Figure 7 shows the new configuration designed to incorporate SRAM. The system is split into two parts. The 80386 motherboard is the system outlined in this section. It consists of the 80386, memory, and bus control logic. The external expansion bus consists of the I/O Channel and the Peripheral Bus. This is the AT Bus and is made available by an expansion connector. The two major changes are the use of static versus dynamic memory and the move of the EPROM and static memory to the local data bus to give improved ROM performance.

Chips and Technology publishes a schematic for a complete AT system based on their chip set. The schematics are based on bipolar logic and use dynamic memory.[17] I modified Chips and Technology's basic design.

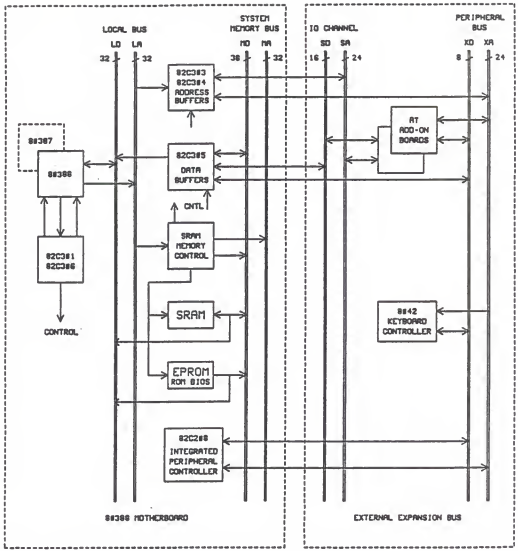

Figure 7: New Chips and Technology Design

4.1.1 Design of 82C302 Replacement

My modification involved designing a SRAM controller to replace the DRAM controller. Since I know the memory configuration for my system and because no refresh logic is needed, my controller is much simpler than the Chips and Technology one. Figure 8 shows the basic controller designed to replace the 82C302. The input signal descriptions are detailed in Table 6. Table 7 lists only the signals that are not required for the former DRAM interface. The signals for the DRAM do not have to be generated and this makes the controller trivial.

Table 6: Memory Control Inputs

Signal	Description
A31-A2	The address bus or part of it to determine if the SRAM or EPROM is requested.
ADS'	80386 ADS' signal to signify a memory cycle.
CLK2	32 MHz clock.
M/IO'	From the 80386 to signify a memory operation.
R/W'	80386 read/write signal.
RESET	This is the reset signal from the 82C301.
WAIT	Wait signal from memory module to extend memory cycle duration.
XMEMR'	This signal indicates that a memory read is to occur for DMA.
XMEMW'	This signal indicates that a memory write is to occur for DMA.

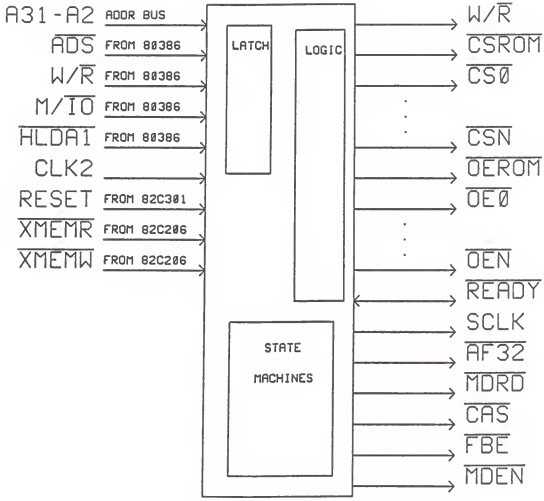

Figure 8: Memory Controller to Replace 82C302.

Table 7: Memory Controller Signals Output

Signal	Description
AF32'	An indication for other Chips and Technology chips that the memory accessed is on the motherboard and not located on the peripheral bus.
BDSEL	A board select signal for the memory module.
CAS'	Output to 82C305 to signal it to latch the data bus in its latches.
CSROM'	A chip select signal for the EPROM.
FBE'	This feature is not required in a normal 80386 system and although the 82C302 generates this signal, it is not used.
MDEN'	This signal is always low and is connected to the 82C305.
MDRD'	Complement of the R/W' signal below which is used by the 82C305.
R/W'	A read/write signal which is based on the R/W' input and the XMEMR' and XMEMW' signals.
READY'	READY signal for end of cycle indication. Must be high impedance during non-memory cycles.
SCLK	CLK2 divided by 2. (16 MHz)

The logic for the signals generated by the controller is detailed below. The AF32' signal indicates that the memory access is on the 80386 board and not on the AT bus. It is

AF32' = (address in range of memory on motherboard)'.

The actual equation will depend on the memory map of the system being built.

The BDSEL signal is asserted for a memory cycle which accesses the common memory module. It is

BDSEL = (address on memory module) and (ADS')'.

The CAS' signal is used to latch data into the 82C305s at the end of a memory cycle. Since the latches are transparent, this signal is the same as the READY' signal below.

The CSROM' signal is asserted like the BDSEL signal, but for ROM accesses. It is

CSROM' = ((address in ROM) and (ADS')')'

The READY' signal is the most complex signal in the controller. This signal is used in a wired, three-state OR configuration. Since the READY' signal is externally pulled up to Vcc, the READY signal can be generated by a three-state buffer which always has a low output. The buffer is only enabled to end the current memory cycle.

This signal can be based on the WAIT signal from the memory module. For example if the memory module allows zero wait state operation except for errors and byte writes, the READY' signal can be enabled as

READY'.oe = (second clock cycle after BDSEL
and WAIT')
OR (third clock cycle after BDSEL).

If the memory module always requires at least one wait state, this signal may be calculated as

READY'.oe = (third clock cycle after BDSEL
and WAIT')
OR (fourth clock cycle after BDSEL).

4.1.2 Timing for Common Memory Module Interface

This section determines the maximum zero wait state clock frequency at which the modified Chips and Technology system may run when using the common memory module. The following assumptions have been made.

- The 80386 data setup time is 11 ns minimum and the 80386 READY# setup time is 21 ns minimum.[18]

- All 80386 local data operations bypass the all Chips and Technology chips and directly attach to the 80386. For DMA operations the Chips and Technology chips are used but they have a minimum setup time of 4 ns which is t_{507} in the Chips and Technology literature [19]. Because the Chips and Technology setup times are much shorter they can be dropped from the analysis.

- The WAIT signal is used to generate the READY signal.

If that the READY# signal can be generated in 10 ns, the following equations govern the 80386 clock frequency.

$$(1 / f_{CLK}) = ALE + ALETOWAIT + READYgen + t_{19}$$

$$= 34 + 115.2 + 10 + 21$$

$$= 180.2 \text{ ns minimum}$$

or

$$f_{CLK} = 11 \text{ MHz}.$$

This calculation shows that it may be desirable to use another mode for the 80386 memory timing such as inserting one wait state in every memory operation.

4.2 Clock Speed Control

The 82C301 Bus Controller provides a method to incorporate different clock speeds into the same system.[20] This will allow the 80386 to meet the multiple clock frequency design constraint. CLK2 will be the high speed clock for the 80386. For zero wait state operation, 11 MHz clock, as calculated above. SCLK will be the 8 MHz AT bus clock. ATCLK1 can be a 4 MHz clock for slower, power-convserving operation. Four megahertz is 80386's lowest operating frequency. The 83C301 is software controllable, so no extra hardware will be required to perform the speed switching.

4.3 Conversion to CMOS

Chips and Technology publishes schematics for a system based on their chip set. These schematics outline a system designed with TTL parts. This section outlines CMOS replacements for these TTL parts. Table 8 outlines the chips used in the Chips and Technology schematics and their CMOS replacements. Many of the components in Chip's and Technology's design will not be required for my design. These chips have been noted in the table.

Table 8: CMOS Parts to Replace TTL Parts

****

U1 74ALS245 will be replaced by CMOS 54AC245.
U3 8742, not part of motherboard.
U4 Keyboard support, not used.
U5 Keyboard support, not used.
U6 82C206 already CMOS.
U7 74S153 will be replaced by CMOS 54AC153.
U8 74LS373 will be replaced by CMOS 54AC373.
U9 74LS373 will be replaced by CMOS 54AC373.
U10 Diagnostic display driver, not used.
U11 Diagnostic display driver, not used.
U12 82C301 already CMOS.
U16 Diagnostic display, not used.
U17 Diagnostic display, not used.
U18 EPROM not used (changed) in new configuration.
U19 EPROM not used (changed) in new configuration.
U20 74ALS245 will be replaced by CMOS 54AC245.
U21 EPROM and 80287 support, not used.
U22 EPROM and 80287 support, not used.
U23 82A304 replaced with CMOS 82C304.
U24 82A303 replaced with CMOS 82C303.
U25 82C302, DRAM controller, not used.
U26 80386 microprocessor, already CMOS.
U27 DRAM support, not used.
U28 DRAM support, not used.
U29 DRAM support, not used.
U30 DRAM support, not used.
U31 DRAM support, not used.
U32 DRAM support, not used.
U33 DRAM support, not used.
U34 DRAM support, not used.
U35 DRAM support, not used.
U36 DRAM support, not used.
U37 DRAM support, not used.
U39 Speaker support, not used.
U40 DRAM support, not used.
U41 DRAM support, not used.
U42 Parity detection, not used.
U43 Parity detection, not used.
U44 74ALS04 replaced with CMOS 54AC04.
U45 74F257 replaced with CMOS 54AC257.
U46 82A305 replaced with CMOS 82C305
U47 DRAM support, not used.
U48 DRAM support, not used.
U49 DRAM support, not used.
U50 DRAM support, not used.

```
U51  82A306 replaced with CMOS 82C306.
U52  82A305 replaced with CMOS 82C305.
U53  74F244 replaced with CMOS 54AC244.
U54  80287, not used in new configuration.
U55  74S153 replaced with CMOS 54AC153.

************************************************
```

Table 9 lists the parts required for the system using the common memory module. The CMOS parts are Fairchild FACT devices and typically are faster than the TTL parts, except for the F-type TTL parts. The parts marked with an asterisk (*) were F-type parts and may not perform adequately. I compared Texas Instrument TTL parts against Fairchild FACT parts to determine this.[21,22]

The system will require four EPROMs. Signetics makes fast EPROMs, 27HC641, which can give zero wait state performance for the system.[23]

I have also allowed five PALs to control generation of the READY# signal, and to provide EPROM control and BDSEL logic. This number may be high.

Table 9: Parts List for the CMOS 80386 System Using the Common Memory Module.

U1, U20	54AC245.
U6	82C206.
U7, U55	54AC153.
U8, U9	54AC373.
U12	82C301.
U23	82C304.
U24	82C303.
U26	M80386.
U44	54AC04.
U45	54AC257. *
U46	82C305
U51	82C306.
U52	82C305.
U53	54AC244. *
SRAM	Common Memory Module
EPROM(4)	27HC641
Misc	5 PALS for control of EPROM, BDSEL, READY#. 80387.

* May not be fast enough.

4.4 Power Estimate for the 80386 System

In this section I obtain an estimate for the power required for the CMOS 80386 system using the common memory module. The power required for the FACT devices is calculated with the formula on page 2-10 of the FACT data book.[24] This formula is

$$PD = (C_L + C_{PD})V_{CC}^2 f \; / \; 1000$$

where

C_L is the load capacitance,

C_{PD} is the device power capacitance (given),

V_{CC} is the power supply voltage,

and

f is the frequency of operation.

I have assumed an operating frequency of 8 MHz and a load of 20 pF for all 54ACxxx-type devices.

Each 54AC245 will require

PD = (20 + 45)(25)(8) / 1000

= 13 mW.

Each 54AC153 will require

PD = (20 + 55)(25)(8) / 1000

= 15 mW.

Each 54AC373 will require

PD = (20 + 40)(25)(8) / 1000

= 12 mW.

The 54AC04 will require

PD = (20 + 30)(25)(8) / 1000

= 10 mW.

The 54AC257 will require

PD = (20 + 50)(25)(8) / 1000

= 14 mW.

The 54AC244 will require

PD = (20 + 45)(25)(8) / 1000

= 13 mW.

The Chips and Technology 82C206 Integrated Peripherals Controller requires 30 mA.[25]

$PD = I_{CC}V_{CC}$

= 30(5)

= 150 mW.

The Chips and Technology chip set requires the following.[26]

82C301 requires 40 mA. p. 33.

82C303 requires 100 mA, p. 75.

82C304 requires 1 mA, p. 86.

Each 82C305 requires 0.1 mA, p. 99.

82C306 requires 200 mA, p. 105.

So, the chips in the chip set require

$$I_{CC} = 40 + 100 + 1 + 2(0.1) + 200$$

$$= 341.2 \text{ mA}$$

and

$$PD = I_{CC}V_{CC}$$

$$= (341.2)(5)$$

$$= 1.7 \text{ W.}$$

The M80386 requires 450 mA.[27]

$$PD = 450(5)$$

$$= 2.25 \text{ W}$$

The Signetics EPROMs require 110 mA.[28]

$$PD = 110(5)$$

$$= 550 \text{ mW.}$$

Four EPROMs will require 2.2 watts.

Table 10 gives a power estimate for the system. I do not have power data available for the 80387. The power estimate for the PALS is the 8 MHz estimate from the previous section on the common memory module. It also does not include estimates for the oscillators.

Table 10: Estimated System Power Requirements.

Part	Number Required	Total Power
54AC245	1	13 mW
54AC153	1	15 mW
54AC373	2	24 mW
54AC04	1	10 mW
54AC257	1	14 mW
54AC244	1	13 mW
82C206	1	150 mW
Chip Set	1	1700 mW
M80386	1	2250 mW
EPROM	4	2200 mW
PALs	5	2000 mW
Memory Module	1	7280 mW
TOTAL		16.0 W

5 Summary

A low-power system design has been outlined. The system consists of two parts, an error correcting memory module and the microprocessor motherboard.

The memory module has 512k 23-bit words of static memory and incorporates a modified Hamming Code to allow detection and correction of all single-bit errors and detection of all double-bit errors. The memory module can be used with most 32-bit microprocessors including the 80386, the 32532, and the DSP32C.

The computer's motherboard is based on Chips and Technology's 80386 chip set. However Chips and Technology's chip set had to be modified to use the common memory module. The modifications were needed to allow the use of SRAM memory. Chips and Technology's DRAM controller was replaced with a SRAM controller as outlined in the text. The redesigned Chips and Technology system will run at 11 MHz if zero wait state operation is desired. This frequency is slow because error correction operations and data validation require extra time. The entire system will use approximately 16 watts of power.

It may be possible to increase the system's clock frequency by using a wait state during each memory access. This would allow the 80386 to run faster when memory operations are not being performed and might increase system performance.

Appendix A

PAL Source Listings

```
Name        mmcntl1;
Partno      XXXXX;
Date        03/23/89;
Revision    01;
Designer    Craig A. Robson;
Company     EECE KSU;
Device      p16r4;


/*********************************************************/
/*  The PALs mmcntl1 and mmcntl2 implement the control   */
/*  logic required for the Common Memory Module.         */
/*********************************************************/
/*  Allowable Target Device Types:   p16r4               */
/*********************************************************/

/**  Inputs  **/

Pin 1         = clock      ;    /* Module's clock.  8 MHz. */
Pin 2         = !ERROR     ;    /* ERROR signal from the   */
                                /*  EDC.                   */
Pin 3         = R_W        ;    /* Read/Write signal to    */
                                /*  the module.            */
Pin 4         = BDSEL      ;    /* Board Select signal to  */
                                /*  the module.            */
Pin 5         = !BE0       ;    /* Byte enable.            */
Pin 6         = !BE1       ;    /* Byte enable.            */
Pin 7         = !BE2       ;    /* Byte enable.            */
Pin 8         = !BE3       ;    /* Byte enable.            */

/**  Outputs  **/

Pin 19        = MREAD      ;    /* MREAD/MWRITE signal to  */
                                /*  SRAM and EDC.          */
Pin 18        = !SRAM_OE   ;    /* Output enable signal    */
                                /*  for the SRAMs.         */
Pin 17        = STATE_0    ;    /* Flip-Flop for State     */
                                /*  machine.               */
Pin 13        = WAIT       ;    /* WAIT signal to edge     */
                                /*  connector.             */

/** Declarations and Intermediate Variable Definitions **/

byte_write = !R_W & !(BE0 & BE1 & BE2 & BE3);

STATE_1    = !STATE_0;
```

```
/**  Logic Equations  **/

STATE_0.d  = ERROR # (byte_write & STATE_0);

MREAD = (byte_write & STATE_0) # (R_W & !byte_write &
    STATE_0);

SRAM_OE = (byte_write & STATE_0) # (R_W & !byte_write &
    STATE_0);

WAIT = BDSEL & ( (byte_write & STATE_0) # (ERROR) );
```

```
Name        mmcntl2;
Partno      XXXXX;
Date        03/23/89;
Revision    01;
Designer    Craig A. Robson;
Company     EECE KSU;
Device      p16l8;


/*********************************************************/
/*  The PALs mmcntl1 and mmcntl2 implement the control   */
/*  logic required for the Common Memory Module.         */
/*********************************************************/
/*  Allowable Target Device Types:   p16l8               */
/*********************************************************/

/**  Inputs  **/

Pin 1         = STATE_0    ;    /* From mmcntl1.           */
Pin 2         = !ERROR     ;    /* ERROR signal from the   */
                                /*  EDC.                   */
Pin 3         = R_W        ;    /* Read/Write signal to    */
                                /*  the module.            */
Pin 4         = BDSEL      ;    /* Board Select signal to  */
                                /*  the module.            */
Pin 5         = !BE0       ;    /* Byte enable.            */
Pin 6         = !BE1       ;    /* Byte enable.            */
Pin 7         = !BE2       ;    /* Byte enable.            */
Pin 8         = !BE3       ;    /* Byte enable.            */

/**  Outputs  **/

Pin 19        = !245_OE0   ;    /* Input/Output buffer     */
Pin 18        = !245_OE1   ;    /*  enable signals.        */
Pin 17        = !245_OE2   ;
Pin 16        = !245_OE3   ;

Pin 15        = !EDC_OE0   ;    /* EDC buffer enable       */
Pin 14        = !EDC_OE1   ;    /*  signals.               */
Pin 13        = !EDC_OE2   ;
Pin 12        = !EDC_OE3   ;


/** Declarations and Intermediate Variable Definitions **/

byte_write = !R_W & !(BE0 & BE1 & BE2 & BE3);

STATE_1    = !STATE_0;
```

```
/**  Logic Equations  **/

245_OE0 = BDSEL & ( (byte_write & STATE_0) #
    (byte_write & !STATE_0 & !BE0) );

245_OE1 = BDSEL & ( (byte_write & STATE_0) #
    (byte_write & !STATE_0 & !BE1) );

245_OE2 = BDSEL & ( (byte_write & STATE_0) #
    (byte_write & !STATE_0 & !BE2) );

245_OE3 = BDSEL & ( (byte_write & STATE_0) #
    (byte_write & !STATE_0 & !BE3) );


EDC_OE0 = (!STATE_0 & !byte_write) #
    (!STATE_0 & byte_write & BE0);

EDC_OE1 = (!STATE_0 & !byte_write) #
    (!STATE_0 & byte_write & BE1);

EDC_OE2 = (!STATE_0 & !byte_write) #
    (!STATE_0 & byte_write & BE2);

EDC_OE3 = (!STATE_0 & !byte_write) #
    (!STATE_0 & byte_write & BE3);
```

References

1. M80386 High Performance 32-Bit Microprocessor with Integrated Memory Management, Intel Corporation, Santa Clara, California, 1987.

2. 80386 Hardware Reference Manual, Intel Corporation, Santa Clara, California, 1986.

3. High-Speed CMOS Data Book, Integrated Device Technology, Inc., Santa Clara, California, p. 8-250.

4. High-Speed CMOS Data Book, Integrated Device Technology, Inc., Santa Clara, California, 1988, p. 10-82.

5. High-Speed CMOS Data Book, Integrated Device Technology, Inc., Santa Clara, California, 1988, p. 10-99.

6. CMOS Data Book, Cypress Semiconductor, San Jose, California, 1988, p. 4-7.

7. CMOS Data Book, Cypress Semiconductor, San Jose, California, 1988, p. 4-7.

8. High-Speed CMOS Data Book, Integrated Device Technology, Inc., Santa Clara, California, 1988, p. 8-235.

9. High-Speed CMOS Data Book, Integrated Device Technology, Inc., Santa Clara, California, 1988, p. 4-94.

10. High-Speed CMOS Data Book, Integrated Device Technology, Inc., Santa Clara, California, 1988, p 10-84.

11. High-Speed CMOS Data Book, Integrated Device Technology, Inc., Santa Clara, California, 1988, p. 10-101.

12. CMOS Data Book, Cypress Semiconductor, San Jose, California, 1988, p. 4-10.

13. High-Speed CMOS Data Book, Integrated Device Technology, Inc., 1988, p. 8-248.

14. High-Speed CMOS Data Book, Integrated Device Technology, Inc., Santa Clara, California, 1988, p. 4-96.

15. 82C301, 82C302, 82A303, 82A304, 82A305, 82A306 CS8230: AT/386 CHIPSet, Chips and Technologies, Incorporated, San Jose, California, 1987.

16. 82C206 Integrated Peripherals Controller, Chips and Technologies, Incorporated, Milpitas, California, 1986.

17. DK8230SCC (schematic), Rev C., Chips and Technologies, Mitpitas, California, 1987.

18. M80386 High Performance 32-Bit Microprocessor with Integrated Memory Management, Intel Corporation, Santa Clara, California, 1987, p. 103.

19. 82C301, 82C302, 82A303, 82A304, 82A305, 82A306 CS8230: AT/386 CHIPSet, Chips and Technologies, Incorporated, San Jose, California, 1987, p. 100.

20. 82C301, 82C302, 82A303, 82A304, 82A305, 82A306 CS8230: AT/386 CHIPSet, Chips and Technologies, Incorporated, San Jose, California, 1987, p 25.

21. Fairchild Advanced CMOS Technology Logic Data Book, Fairchild Semiconductor Corporation, South Portland, Maine, 1987.

22. The TTL Data Book, Volume 2, Texas Instruments, Dallas, Texas, 1985.

23. 27HC641, 65,536-Bit UV Erasable CMOS PROM (8K x 8), Product Specification, Signetics, Sunnyvale, California, 1988.

24. Fairchild Advanced CMOS Technology Logic Data Book, Fairchild Semiconductor Corporation, South Portland, Maine, 1987, p. 2-10.

25. 82C206 Integrated Peripherals Controller, Chips and Technology, Incorporated, Milpitas, California, 1986, p. 41.

26. 82C301, 82C302, 82A303, 82A304, 82A305, 82A306 CS8230: AT/386 CHIPSet, Chips and Technologies, Incorporated, San Jose, California, 1987.

27. M80386 High Performance 32-Bit Microprocessor with Integrated Memory Management, Intel Corporation, Santa Clara, California, 1987, p. 101.

28. 27HC641, 65,536-Bit UV Erasable CMOS PROM (8K x 8), Product Specification, Signetics, Sunnyvale, California, 1988, p. 2.

Design of a Low Power 80386 Based System

by

Craig A. Robson

BSEE, Kansas State University, 1987

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment
of the requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1989

ABSTRACT

This thesis outlines the design of a low-power 80386 based computer. The first section outlines the design of a common memory module which incorporates a Modified Hamming Code error detection and correction scheme. This module is designed to work with three different 32-bit microprocessors. The second section outlines the modification of an existing design to use the memory module described in the first section. The entire system is designed using CMOS parts for low power consumption.