CHAPTER 10

Memory Interface

CINMY
SIR

INTRODUCTION

Whether simple or complex, every microprocessor-based system has a memory system. The Intel family of microprocessors is no different from any other in this respect. Almost all systems contain two main types of memory: **read-only memory** (ROM) and **random access memory** (RAM) or read/write memory. Read-only memory contains system software and permanent system data, while RAM contains temporary data and application software. This chapter explains how to interface both memory types to the Intel family of microprocessors. We demonstrate memory interface to an 8-, 16-, 32-, and 64-bit data bus by using various memory address sizes. This allows virtually any microprocessor to be interfaced to any memory system.

CHAPTER OBJECTIVES

Upon completion of this chapter, you will be able to:

1. Decode the memory address and use the outputs of the decoder to select various memory components.
2. Use programmable logic devices (PLDs) to decode memory addresses.
3. Explain how to interface both RAM and ROM to a microprocessor.
4. Explain how error correction code (ECC) is used with memory.
5. Interface memory to an 8-, 16-, 32-, and 64-bit data bus.
6. Explain the operation of a dynamic RAM controller.
7. Interface dynamic RAM to the microprocessor.

10–1 MEMORY DEVICES

Before attempting to interface memory to the microprocessor, it is essential to completely understand the operation of memory components. In this section, we explain the functions of the four common types of memory: read-only memory (ROM), flash memory (EEPROM), static random access memory (SRAM), and dynamic random access memory (DRAM).

Memory Pin Connections

Pin connections common to all memory devices are the address inputs, data outputs or input/outputs, some type of selection input, and at least one control input used to select a read or write operation. See Figure 10–1 for ROM and RAM generic-memory devices.

Address Connections. All memory devices have address inputs that select a memory location within the memory device. Address inputs are almost always labeled from A_0, the least significant address input, to A_n where subscript n can be any value but is always labeled as one less than the total number of address pins. For example, a memory device with 10 address pins has its address pins labeled from A_0 to A_9. The number of address pins found on a memory device is determined by the number of memory locations found within it.

Today, the more common memory devices have between 1K (1024) to 1G (1,073,741,824) memory locations, with 4G and larger memory location devices on the horizon. A 1K memory device has 10 address pins (A_0–A_9); therefore, 10 address inputs are required to select any of its 1024 memory locations. It takes a 10-bit binary number (1024 different combinations) to select any single location on a 1024-location device. If a memory device has 11 address connections (A_0–A_{11}), it has 2048 (2K) internal memory locations. The number of memory locations can thus be extrapolated from the number of address pins. For example, a 4K memory device has 12 address connections, an 8K device has 13, and so forth. A device that contains 1M locations requires a 20-bit address (A_0–A_{19}).

The number 400H represents a 1K-byte section of the memory system. If a memory device is decoded to begin at memory address 10000H and it is a 1K device, its last location is at address 103FFH—one location less than 400H. Another important hexadecimal number to remember is 1000H, because 1000H is 4K. A memory device that contains a starting address of 14000H that is 4K bytes long ends at location 14FFFH—one location less than 1000H. A third number is 64K, or 10000H. A memory that starts at location 30000H and ends at location 3FFFFH is a 64K-byte memory. Finally, because 1M of memory is common, a 1M memory contains 100000H memory locations.

Data Connections. All memory devices have a set of data outputs or input/outputs. The device illustrated in Figure 10–1 has a common set of input/output (I/O) connections. Today, many memory devices have bidirectional common I/O pins.

The data connections are the points at which data are entered for storage or extracted for reading. Data pins on memory devices are labeled D_0 through D_7 for an 8-bit-wide memory

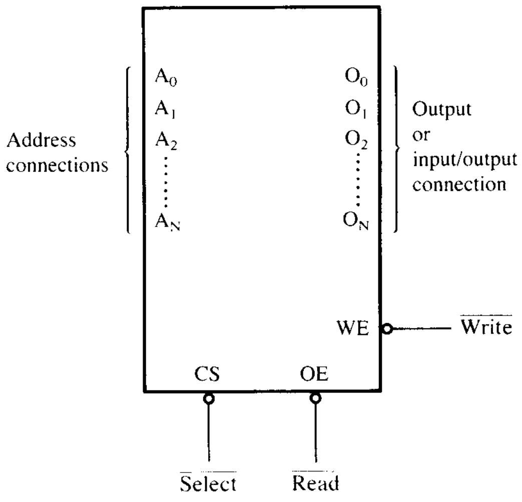


FIGURE 10–1 A pseudo-memory component illustrating the address, data, and control connections.

device. In this sample memory device there are 8 I/O connections, meaning that the memory device stores 8 bits of data in each of its memory locations. An 8-bit-wide memory device is often called a **byte-wide** memory. Although most devices are currently 8 bits wide, some devices are 16 bits, 4 bits, or just 1 bit wide.

Catalog listings of memory devices often refer to memory locations times bits per location. For example, a memory device with 1K memory locations and 8 bits in each location is often listed as a $1K \times 8$ by the manufacturer. A $16K \times 1$ is a memory device containing 16K 1-bit memory locations. Memory devices are often classified according to total bit capacity. For example, a $1K \times 8$-bit memory device is sometimes listed as an 8K memory device, or a $64K \times 4$ memory is listed as a 256K device. These variations occur from one manufacturer to another.

Selection Connections. Each memory device has an input—sometimes more than one—that selects or enables the memory device. This type of input is most often called a **chip select** ($\overline{CS}$), **chip enable** ($\overline{CE}$), or simply **select** ($\overline{S}$) input. RAM memory generally has at least one $\overline{CS}$ or $\overline{S}$ input, and ROM has at least one $\overline{CE}$. If the $\overline{CE}$, $\overline{CS}$, or $\overline{S}$ input is active (a logic 0, in this case, because of the overbar), the memory device performs a read or write operation; if it is inactive (a logic 1, in this case), the memory device cannot do a read or a write because it is turned off or disabled. If more than one $\overline{CS}$ connection is present, all must be activated to read or write data.

Control Connections. All memory devices have some form of control input or inputs. A ROM usually has only one control input, while a RAM often has one or two control inputs.

The control input most often found on a ROM is the **output enable** ($\overline{OE}$) or **gate** ($\overline{G}$) connection, which allows data to flow out of the output data pins of the ROM. If $\overline{OE}$ and the selection input ($\overline{CE}$) are both active, the output is enabled; if $\overline{OE}$ is inactive, the output is disabled at its high-impedance state. The $\overline{OE}$ connection enables and disables a set of three-state buffers located within the memory device and must be active to read data.

A RAM memory device has either one or two control inputs. If there is only one control input, it is often called $R/\overline{W}$. This pin selects a read operation or a write operation only if the device is selected by the selection input ($\overline{CS}$). If the RAM has two control inputs, they are usually labeled $\overline{WE}$ (or $\overline{W}$), and $\overline{OE}$ (or $\overline{G}$). Here, $\overline{WE}$ (**write enable**) must be active to perform a memory write, and $\overline{OE}$ must be active to perform a memory read operation. When these two controls ($\overline{WE}$ and $\overline{OE}$) are present, they must never both be active at the same time. If both control inputs are inactive (logic 1s), data are neither written nor read, and the data connections are at their high-impedance state.

ROM Memory

Self Study

The **read-only memory** (ROM) permanently stores programs and data that are resident to the system and must not change when power supply is disconnected. The ROM is permanently programmed so that data are always present, even when power is disconnected. This type of memory is often called **nonvolatile memory**, because its contents *do not* change even if power is disconnected.

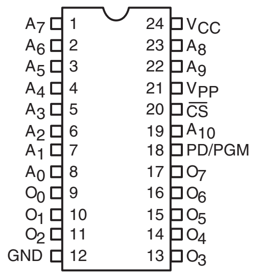
Today, the ROM is available in many forms. A device we call a ROM is purchased in mass quantities from a manufacturer and programmed during its fabrication at the factory. The EPROM (**erasable programmable read-only memory**), a type of ROM, is more commonly used when software must be changed often or when too few are in demand to make the ROM economical. For a ROM to be practical, we usually must purchase at least 10,000 devices to recoup the factory programming charge. An EPROM is programmed in the field on a device called an EPROM programmer. The EPROM is also erasable if exposed to high-intensity ultraviolet light for about 20 minutes or so, depending on the type of EPROM.

PROM memory devices are also available, although they are not as common today. The PROM (**programmable read-only memory**) is also programmed in the field by burning open tiny NI-chrome or silicon oxide fuses; but once it is programmed, it cannot be erased.

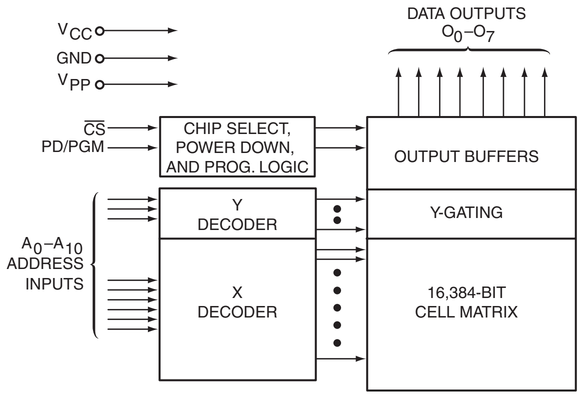
MODE SELECTION

MODE \ PINS	PD/PGM (18)	$\overline{CS}$ (20)	V_{PP} (21)	V_{CC} (24)	OUTPUTS (9-11, 13-17)
Read	V_{IL}	V_{IL}	+5	+5	D_{OUT}
Deselect	Don't care	V_{IH}	+5	+5	High Z
Power Down	V_{IH}	Don't care	+5	+5	High Z
Program	Pulsed V_{IL} to V_{IH}	V_{IH}	+25	+5	D_{IN}
Program Verify	V_{IL}	V_{IL}	+25	+5	D_{OUT}
Program Inhibit	V_{IL}	V_{IH}	+25	+5	High Z

PIN CONFIGURATION



BLOCK DIAGRAM



PIN NAMES

A_0–A_{10}	ADDRESSES
PD/PGM	POWER DOWN/PROGRAM
$\overline{CS}$	CHIP SELECT
O_0–O_7	OUTPUTS

FIGURE 10–2 The pin-out of the 2716, 2K × 8 EPROM. (Courtesy of Intel Corporation.)

Still another, newer type of **read-mostly memory** (RMM) is called the **flash memory**. The flash memory[1] is also often called an EEPROM (**electrically erasable programmable ROM**), EAROM (**electrically alterable ROM**), or a NOVRAM (**nonvolatile RAM**). These memory devices are electrically erasable in the system, but they require more time to erase than a normal RAM. The flash memory device is used to store setup information for systems such as the video card in the computer. It has all but replaced the EPROM in most computer systems for the BIOS memory. Some systems contain a password stored in the flash memory device. Flash memory has its biggest impact in memory cards for digital cameras and memory in MP3 audio players.

Figure 10–2 illustrates the 2716 EPROM, which is representative of most common EPROMs. This device contains 11 address inputs and eight data outputs. The 2716 is a 2K × 8 read-only memory device. The 27XXX series of the EPROMs includes the following part numbers: 2704 (512 × 8), 2708 (1K × 8), 2716 (2K × 8), 2732 (4K × 8), 2764 (8K × 8), 27128 (16K × 8), 27256 (32K × 8), 27512 (64K × 8), and 271024 (128K × 8). Each of these parts contains address pins, eight data connections, one or more chip selection inputs ($\overline{CE}$), and an output enable pin ($\overline{OE}$).

Figure 10–3 illustrates the timing diagram for the 2716 EPROM. Data appear on the output connections only after a logic 0 is placed on both $\overline{CE}$ and $\overline{OE}$ pin connections. If $\overline{CE}$ and $\overline{OE}$ are not both logic 0s, the data output connections remain at their high-impedance or off states. Note that the V_{PP} pin must be placed at a logic 1 level for data to be read from the EPROM. In some cases, the V_{PP} pin is in the same position as the $\overline{WE}$ pin on the SRAM. This will allow a single socket to hold either an EPROM or an SRAM. An example is the 27256 EPROM and the 62256 SRAM, both 32K × 8 devices that have the same pin-out, except for V_{PP} on the EPROM and $\overline{WE}$ on the SRAM.

[1]Flash memory is a trademark of Intel Corporation.

A.C. Characteristics

$T_A = 0°C$ to $70°C$, V_{CC}[1] = +5V ±5%, V_{PP}[2] = V_{CC} ±0.6V[3]

Symbol	Parameter	Limits			Unit	Test Conditions
		Min.	Typ.[4]	Max.		
t_{ACC1}	Address to Output Delay		250	450	ns	PD/PGM = $\overline{CS}$ = V_{IL}
t_{ACC2}	PD/PGM to Output Delay		280	450	ns	$\overline{CS}$ = V_{IL}
t_{CO}	Chip Select to Output Delay			120	ns	PD/PGM = V_{IL}
t_{PF}	PD/PGM to Output Float	0		100	ns	$\overline{CS}$ = V_{IL}
t_{DF}	Chip Deselect to Output Float	0		100	ns	PD/PGM = V_{IL}
t_{OH}	Address to Output Hold	0			ns	PD/PGM = $\overline{CS}$ = V_{IL}

Capacitance[5] $T_A = 25°C$, f = 1 MHz

Symbol	Parameter	Typ.	Max.	Unit	Conditions
C_{IN}	Input Capacitance	4	6	pF	V_{IN} = 0V
C_{OUT}	Output Capacitance	8	12	pF	V_{OUT} = 0V

A.C. Test Conditions:

Output Load: 1 TTL gate and C_L = 100 pF
Input Rise and Fall Times: ≤20 ns
Input Pulse Levels: 0.8V to 2.2V
Timing Measurement Reference Level:
Inputs 1V and 2V
Outputs 0.8V and 2V

WAVEFORMS

A. Read Mode
PD/PGM = V_{IL}

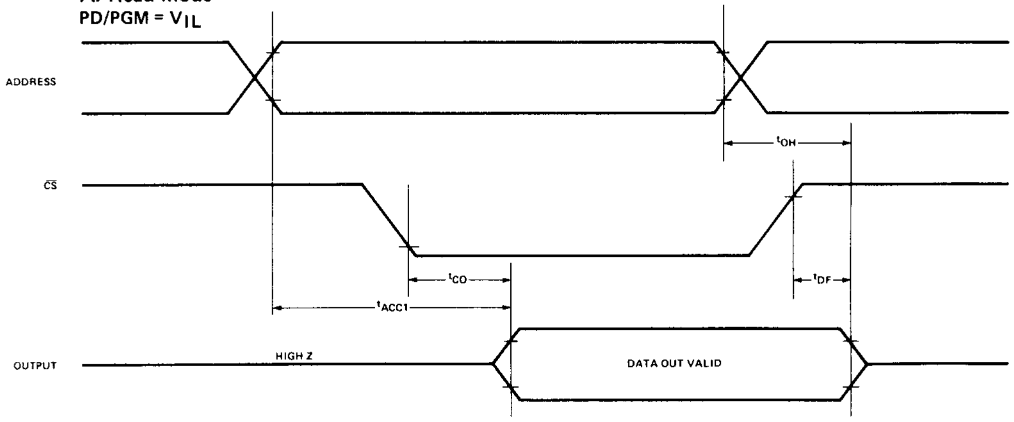


FIGURE 10–3 The timing diagram of AC characteristics of the 2716 EPROM. (Courtesy of Intel Corporation.)

One important piece of information provided by the timing diagram and data sheet is the memory access time—the time that it takes the memory to read information. As Figure 10–3 illustrates, memory access time (T_{ACC}) is measured from the appearance of the address at the address inputs until the appearance of the data at the output connections. This is based on the assumption that the $\overline{CE}$ input goes low at the same time that the address inputs become stable. Also, $\overline{OE}$ must be a logic 0 for the output connections to become active. The basic speed of this EPROM is 450 ns. (Recall that the 8086/8088 operated with a 5 MHz clock allowed memory 460 ns to access data.) This type of memory component requires wait states to operate properly with the 8086/8088 microprocessors because of its rather long access time. If wait states are not desired, higher-speed versions of the EPROM are available at an additional cost. Today, EPROM memory is available with access times of as little as 100 ns. Obviously, wait states are required in modern microprocessors for any EPROM device.

Static RAM (SRAM) Devices

Static RAM memory devices retain data for as long as DC power is applied. Because no special action (except power) is required to retain stored data, these devices are called **static memory**. They are also called **volatile memory** because they will not retain data without power. The main

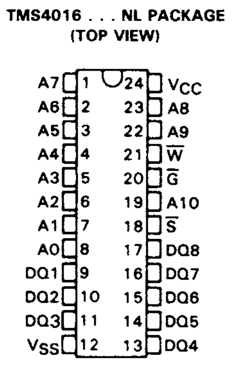


FIGURE 10–4 The pin-out of the TMS4016, 2K × 8 static RAM (SRAM). (Courtesy of Texas Instruments Incorporated.)

PIN NOMENCLATURE	
A0 – A10	Addresses
DQ1 – DQ8	Data In/Data Out
$\overline{G}$	Output Enable
$\overline{S}$	Chip Select
V_{CC}	+5-V Supply
V_{SS}	Ground
$\overline{W}$	Write Enable

difference between a ROM and a RAM is that a RAM is written under normal operation, whereas a ROM is programmed outside the computer and normally is only read. The SRAM, which stores temporary data, is used when the size of the read/write memory is relatively small. Today, a small memory is one that is less than 1M byte.

Figure 10–4 illustrates the 4016 SRAM, which is a 2K × 8 read/write memory. This device has 11 address inputs and eight data input/output connections. This device is representative of all SRAM devices, except for the number of address and data connections.

The control inputs of this RAM are slightly different from those presented earlier. The $\overline{OE}$ pin is labeled $\overline{G}$, the $\overline{CS}$ pin is $\overline{S}$, and the $\overline{WE}$ pin is $\overline{W}$. Despite the altered designations, the control pins function exactly the same as those outlined previously. Other manufacturers make this popular SRAM under the part numbers 2016 and 6116.

Figure 10–5 depicts the timing diagram for the 4016 SRAM. As the read cycle timing reveals, the access time is $t_{a(A)}$. On the slowest version of the 4016, this time is 250 ns, which is fast enough to connect directly to an 8088 or an 8086 operated at 5 MHz without wait states. Again, it is important to remember that the access time must be checked to determine the compatibility of memory components with the microprocessor.

Figure 10–6 on p. 336 illustrates the pin-out of the 62256, 32K × 8 static RAM. This device is packaged in a 28-pin integrated circuit and is available with access times of 120 ns or 150 ns. Other common SRAM devices are available in 8K × 8, 128K × 8, 256K × 8, 512K × 8, and 1M × 8 sizes, with access times of as little as 1.0 ns for SRAM used in computer cache memory systems.

Dynamic RAM (DRAM) Memory

About the largest static RAM available today is a 1M × 8. Dynamic RAM, on the other hand, is available in much larger sizes: up to 256M × 8 (2G bits). In all other respects, DRAM is essentially the same as SRAM, except that it retains data for only 2 or 4 ms on an integrated capacitor. After 2 or 4 ms, the contents of the DRAM must be completely rewritten *(refreshed)* because the capacitors, which store a logic 1 or logic 0, lose their charges.

Instead of requiring the almost impossible task of reading the contents of each memory location with a program and then rewriting them, the manufacturer has internally constructed the DRAM differently from the SRAM. In the DRAM, the entire contents of the memory are refreshed with 256 reads in a 2- or 4-ms interval. Refreshing also occurs during a write, a read, or during a special refresh cycle. Much more information about DRAM refreshing is provided in Section 10–6.

Another disadvantage of DRAM memory is that it requires so many address pins that the manufacturers have decided to multiplex the address inputs. Figure 10–7 illustrates a 64K × 4 DRAM, the TMS4464, which stores 256K bits of data. Notice that it contains only eight address inputs where it should contain 16—the number required to address 64K memory locations. The

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High level voltage	$I_{OH} = -1$ mA,	$V_{CC} = 4.5$ V	2.4			V
V_{OL}	Low level voltage	$I_{OL} = 2.1$ mA,	$V_{CC} = 4.5$ V			0.4	V
I_I	Input current	$V_I = 0$ V to 5.5 V				10	µA
I_{OZ}	Off-state output current	$\overline{S}$ or $\overline{G}$ at 2 V or $\overline{W}$ at 0.8 V, $V_O = 0$ V to 5.5 V				10	µA
I_{CC}	Supply current from V_{CC}	$I_O = 0$ mA, $T_A = 0°C$ (worst case)	$V_{CC} = 5.5$ V,		40	70	mA
C_i	Input capacitance	$V_I = 0$ V,	f = 1 MHz			8	pF
C_O	Output capacitance	$V_O = 0$ V,	f = 1 MHz			12	pF

†All typical values are at $V_{CC} = 5$ V, $T_A = 25°C$.

timing requirements over recommended supply voltage range and operating free-air temperature range

PARAMETER		TMS4016-12		TMS4016-15		TMS4016-20		TMS4016-25		UNIT
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$t_{c(rd)}$	Read cycle time	120		150		200		250		ns
$t_{c(wr)}$	Write cycle time	120		150		200		250		ns
$t_{w(W)}$	Write pulse width	60		80		100		120		ns
$t_{su(A)}$	Address setup time	20		20		20		20		ns
$t_{su(S)}$	Chip select setup time	60		80		100		120		ns
$t_{su(D)}$	Data setup time	50		60		80		100		ns
$t_{h(A)}$	Address hold time	0		0		0		0		ns
$t_{h(D)}$	Data hold time	5		10		10		10		ns

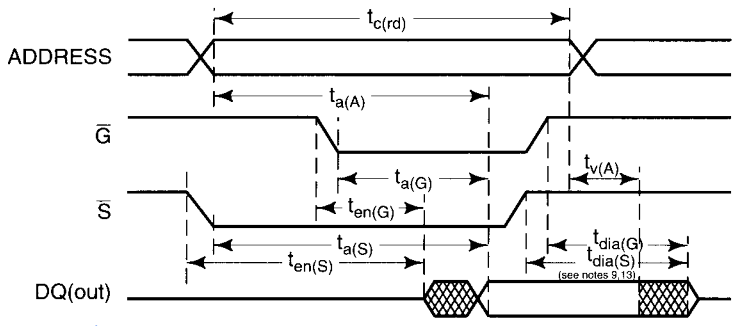
switching characteristics over recommended voltage range, $T_A = 0°C$ to $70°C$

PARAMETER		TMS4016-12		TMS4016-15		TMS4016-20		TMS4016-25		UNIT
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$t_{a(A)}$	Access time from address		120		150		200		250	ns
$t_{a(S)}$	Access time from chip select low		60		75		100		120	ns
$t_{a(G)}$	Access time from output enable low		50		60		80		100	ns
$t_{v(A)}$	Output data valid after address change	10		15		15		15		ns
$t_{dis(S)}$	Output disable time after chip select high		40		50		60		80	ns
$t_{dis(G)}$	Output disable time after output enable high		40		50		60		80	ns
$t_{dis(W)}$	Output disable time after write enable low		50		60		60		80	ns
$t_{en(S)}$	Output enable time after chip select low	5		5		10		10		ns
$t_{en(G)}$	Output enable time after output enable low	5		5		10		10		ns
$t_{en(W)}$	Output enable time after write enable high	5		5		10		10		ns

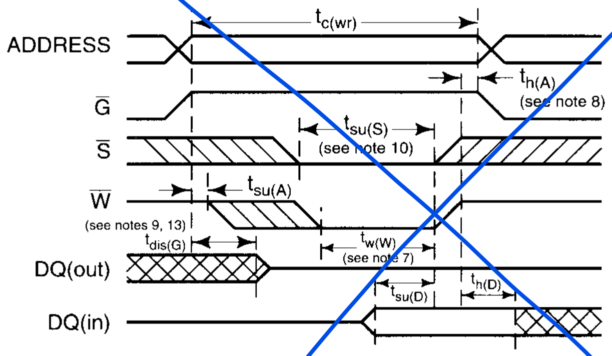
NOTES: 3. $C_L = 100$pF for all measurements except $t_{dis(W)}$ and $t_{en(W)}$.
$C_L = 5$ pF for $t_{dis(W)}$ and $t_{en(W)}$.
4. t_{dis} and t_{en} parameters are sampled and not 100% tested.

FIGURE 10–5 (a) The AC characteristics of the TMS4016 SRAM. (b) The timing diagrams of the TMS4016 SRAM. (Courtesy of Texas Instruments Incorporated.)

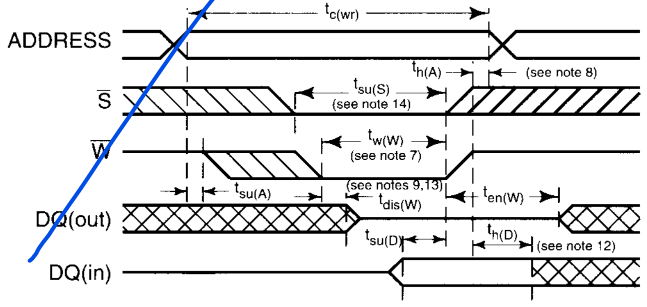
timing waveform of read cycle (see note 5)



timing waveform of write cycle no. 1 (see note 6)



timing waveform of write cycle no. 2 (see notes 6 and 11)



NOTES: 5. $\overline{W}$ is high Read Cycle.
6. $\overline{W}$ must be high during all address transitions.
7. A write occurs during the overlap of a low $\overline{S}$ and a low $\overline{W}$.
8. $t_{h(A)}$ is measured from the earlier of $\overline{S}$ or $\overline{W}$ going high to the end of the write cycle.
9. During this period, I/O pins are in the output state so that the input signals of opposite phase to the outputs must not be applied.
10. If the Slow transition occurs simultaneously with the $\overline{W}$ low transitions or after the $\overline{W}$ transition, output remains in a high impedance state.
11. G is continuously low (G = V_{IL}).
12. If $\overline{S}$ is low during this period, I/O pins are in the output state. Data input signals of opposite phase to the outputs must not be applied.
13. Transition is measured ± 200 mV from steady-state voltage.
14. If the $\overline{S}$ low transition occurs before the W low transition, then the data input signals of opposite phase to the outputs must not be applied for the duration of $t_{dis(W)}$ after the $\overline{W}$ low transition.

FIGURE 10–5 *(continued)*

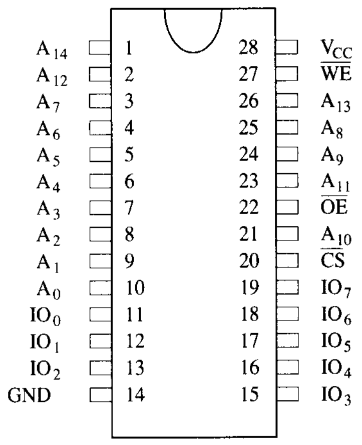
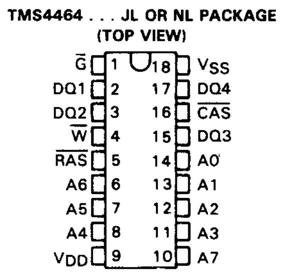


FIGURE 10–6 Pin diagram of the 62256, 32K × 8 static RAM.

PIN FUNCTION

A_0 - A_{14}	Addresses
IO_0 - IO_7	Data connections
$\overline{CS}$	Chip select
$\overline{OE}$	Output enable
$\overline{WE}$	Write enable
V_{CC}	+5V Supply
GND	Ground



(a)

PIN NOMENCLATURE	
A0-A7	Address Inputs
$\overline{CAS}$	Column Address Strobe
DQ1-DQ4	Data-In/Data-Out
$\overline{G}$	Output Enable
$\overline{RAS}$	Row Address Strobe
V_{DD}	+5-V Supply
V_{SS}	Ground
$\overline{W}$	Write Enable

(b)

FIGURE 10–7 The pin-out of the TMS4464, 64K × 4 dynamic RAM (DRAM). (Courtesy of Texas Instruments Incorporated.)

only way that 16 address bits can be forced into eight address pins is in two 8-bit increments. This operation requires two special pins: the **column address strobe** ($\overline{CAS}$) and **row address strobe** ($\overline{RAS}$). First, A_0–A_7 are placed on the address pins and strobed into an internal row latch by $\overline{RAS}$ as the row address. Next, the address bits A_8-A_{15} are placed on the same eight address inputs and strobed into an internal column latch by $\overline{CAS}$ as the column address (see Figure 10–8 for this timing). The 16-bit address held in these internal latches addresses the contents of one of the 4-bit memory locations. Note that $\overline{CAS}$ also performs the function of the chip selection input to the DRAM.

Figure 10–9 illustrates a set of multiplexers used to strobe the column and row addresses into the eight address pins on a pair of TMS4464 DRAMs. Here, the $\overline{RAS}$ signal not only strobes

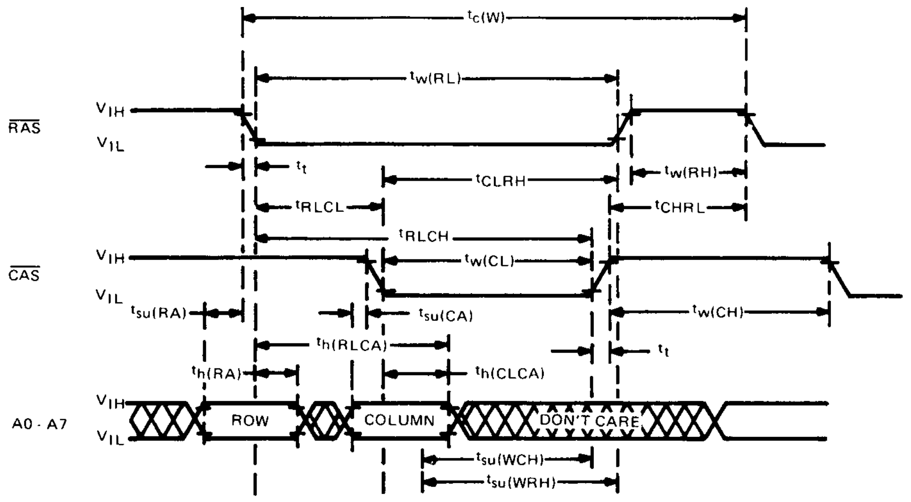


FIGURE 10–8 $\overline{RAS}$, $\overline{CAS}$, and address input timing for the TMS4464 DRAM. (Courtesy of Texas Instruments Incorporated.)

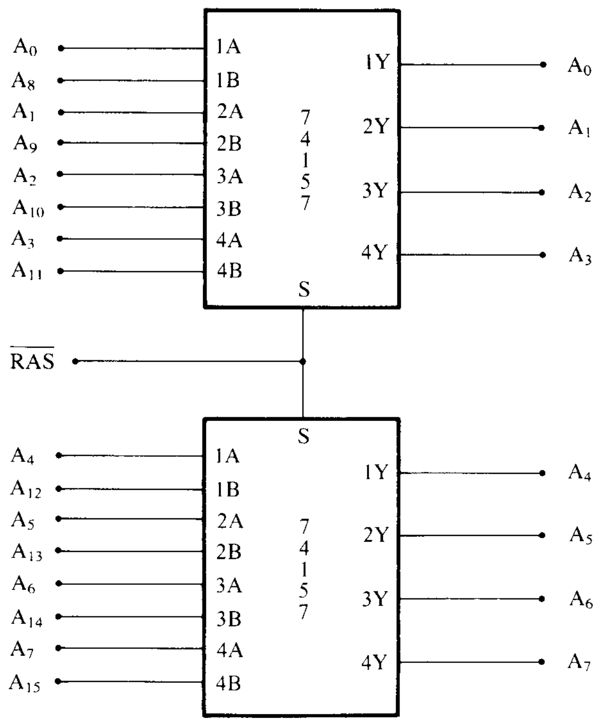


FIGURE 10–9 Address multiplexer for the TMS4464 DRAM.

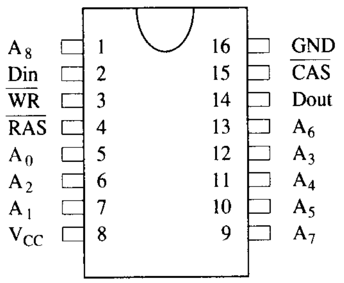


FIGURE 10–10 The 41256 dynamic RAM organized as a 256K × 1 memory device.

PIN FUNCTIONS

A_0 - A_8	Addresses
Din	Data in
Dout	Data out
$\overline{\text{CAS}}$	Column Address Strobe
$\overline{\text{RAS}}$	Row Address Strobe
$\overline{\text{WR}}$	Write enable
V_{CC}	+5V Supply
GND	Ground

the row address into the DRAMs, but it also selects which part of the address is applied to the address inputs. This is possible due to the long propagation-delay time of the multiplexers. When $\overline{\text{RAS}}$ is a logic 1, the B inputs are connected to the Y outputs of the multiplexers; when the $\overline{\text{RAS}}$ input goes to a logic 0, the A inputs connect to the Y outputs. Because the internal row address latch is edge-triggered, it captures the row address before the address at the inputs changes to the column address. More detail on DRAM and DRAM interfacing is provided in Section 10–6.

As with the SRAM, the $R/\overline{W}$ pin writes data to the DRAM when a logic 0, but there is no pin labeled G or enable. There also is no $\overline{S}$ (select) input to the DRAM. As mentioned, the $\overline{\text{CAS}}$ input selects the DRAM. If selected, the DRAM is written if $R/\overline{W} = 0$ and read if $R/\overline{W} = 1$.

Figure 10–10 shows the pin-out of the 41256 dynamic RAM. This device is organized as a 256K × 1 memory, requiring as little as 70 ns to access data.

More recently, larger DRAMs have become available that are organized as a 16M × 1, 256M × 1, 512M × 1, 1G × 1, and 2G × 1 memory. On the horizon is the 4G × 1 memory, which is in the planning stages. DRAM memory is often placed on small circuit boards called SIMMs (Single In-Line Memory Modules). Figure 10–11 shows the pin-outs of the two most common SIMMs. The 30-pin SIMM is organized most often as 1M × 8 or 1M × 9, and 4M × 8 or 4M × 9. (Illustrated in Figure 10–11 is a 4M × 9.) The ninth bit is the parity bit. Also shown is a newer 72 pin SIMM. The 72-pin SIMMs are often organized as 1M × 32 or 1M × 36 (with parity). Other sizes are 2M × 32, 4M × 32, 8M × 32, and 16M × 32. These are also available with parity. Figure 10–11 illustrates a 4M × 36 SIMM, which has 16M bytes of memory.

Lately, many systems are using the Pentium–Pentium 4 microprocessors. These microprocessors have a 64-bit wide data bus, which precludes the use of the 8-bit-wide SIMMs described here. Even the 72-pin SIMMs are cumbersome to use because they must be used in pairs to obtain a 64-bit-wide data connection. Today, the 64-bit-wide DIMMs (Dual In-line Memory Modules) have become the standard in most systems. The memory on these modules is organized as 64 bits wide. The common sizes available are 16M bytes (2M × 64), 32M bytes (4M × 64), 64M bytes (8M × 64), 128M bytes (16M × 64), 256M bytes (32M × 64), 512M bytes (64M × 64), and 1G bytes (128M × 64). The pin-out of the DIMM is illustrated in Figure 10–12. The DIMM module is available in DRAM, EDO, SDRAM, and DDR (double-data rate) forms,

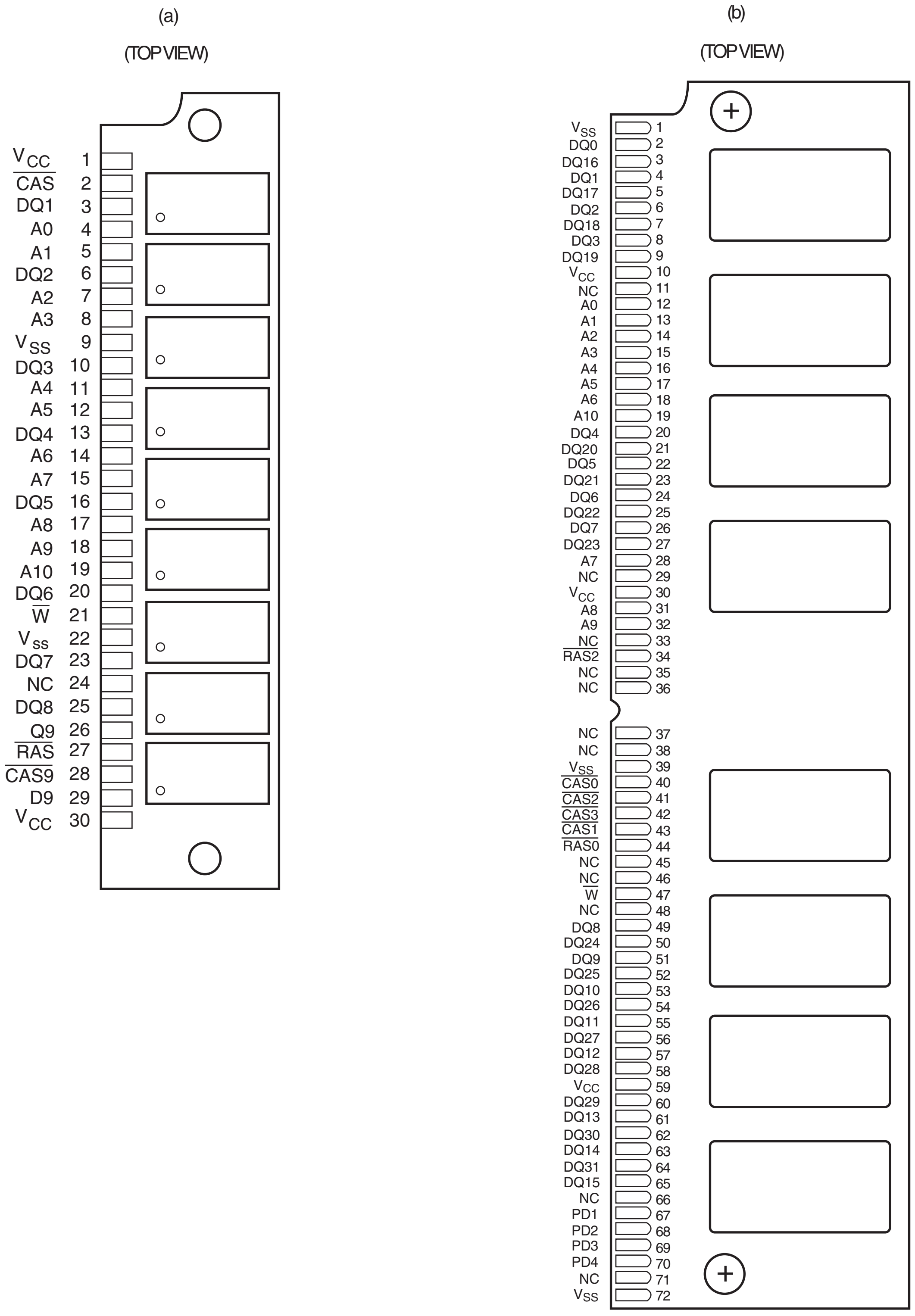


FIGURE 10–11 The pin-outs of the 30-pin and 72-pin SIMM. (a) A 30-pin SIMM organized as 4M × 9 and (b) a 72-pin SIMM organized as 4M × 36.

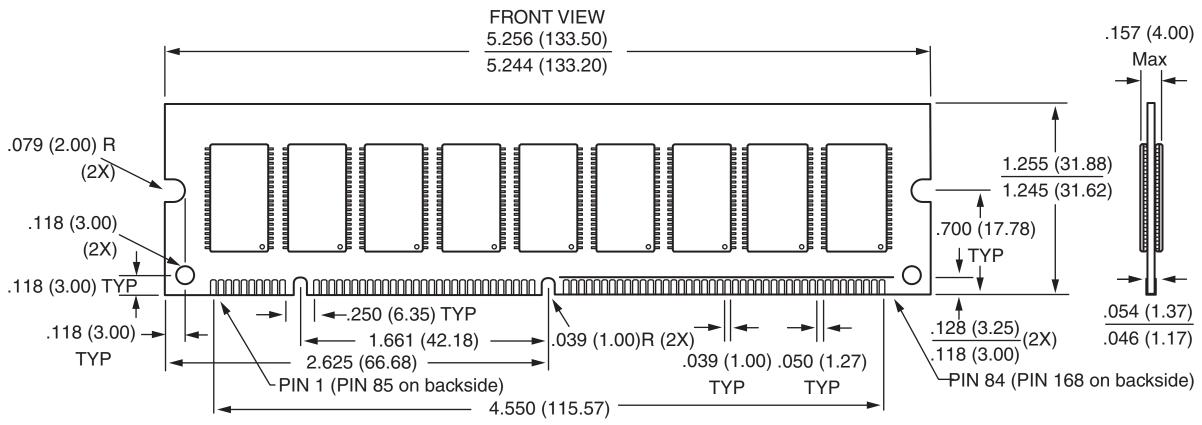


FIGURE 10–12 The pin-out of a 168-pin DIMM.

with or without an EPROM. The EPROM provides information to the system on the size and the speed of the memory device for plug-and-play applications.

Another addition to the memory market is the RIMM memory module from RAMBUS Corporation, although this memory type has faded from the market. Like the SDRAM, the RIMM contains 168 pins, but each pin is a two-level pin, bringing the total number of connections to 336. The fastest SDRAM currently available is the PC-4400 or 500 DDR, which operates at a rate of 4.4G bytes per second. By comparison, the 800 MHz RIMM operates at a rate of 3.2G bytes per second and is no longer supported in many systems. RDRAM had a fairly short life in the volatile computer market. The RIMM module is organized as a 32-bit-wide device. This means that to populate a Pentium 4 memory, RIMM memory is used in pairs. Intel claims that the Pentium 4 system using RIMM modules is 300% faster than a Pentium III using PC-100 memory. According to RAMBUS, the current 800 MHz RIMM has been increased to a speed of 1200 MHz, but it is still not fast enough to garner much of a market share.

Currently the latest DRAM is the DDR (**double-data rate**) memory device and DDR2. The DDR memory transfers data at each edge of the clock, making it operate at twice the speed of SDRAM. This does not affect the access time for the memory, so many wait states are still required to operate this type of memory, but it can be much faster than normal SDRAM memory and that includes RDRAM.

10–2 ADDRESS DECODING

In order to attach a memory device to the microprocessor, it is necessary to decode the address sent from the microprocessor. Decoding makes the memory function at a unique section or partition of the memory map. Without an address decoder, only one memory device can be connected to a microprocessor, which would make it virtually useless. In this section, we describe a few of the more common address-decoding techniques, as well as the decoders that are found in many systems.

Why Decode Memory?

When the 8088 microprocessor is compared to the 2716 EPROM, a difference in the number of address connections is apparent—the EPROM has 11 address connections and the microprocessor has 20. This means that the microprocessor sends out a 20-bit memory address whenever it reads or writes data. Because the EPROM has only 11 address pins, there is a mismatch that must be corrected. If only 11 of the 8088's address pins are connected to the memory, the 8088 will see only 2K bytes of memory instead of the 1M bytes that it "expects" the memory to contain. The decoder corrects the mismatch by decoding the address pins that do not connect to the memory component.

Simple NAND Gate Decoder

When the 2K × 8 EPROM is used, address connections A_{10}–A_0 of the 8088 are connected to address inputs A_{10}–A_0 of the EPROM. The remaining nine address pins (A_{19}–A_{11}) are connected to the inputs of a NAND gate decoder (see Figure 10–13). The decoder selects the EPROM from one of the 2K-byte sections of the 1M-byte memory system in the 8088 microprocessor.

In this circuit, a single NAND gate decodes the memory address. The output of the NAND gate is a logic 0 whenever the 8088 address pins attached to its inputs (A_{19}–A_{11}) are all logic 1s. The active low, logic 0 output of the NAND gate decoder is connected to the $\overline{\text{CE}}$ input pin that selects (enables) the EPROM. Recall that whenever $\overline{\text{CE}}$ is a logic 0, data will be read from the EPROM only if $\overline{\text{OE}}$ is also a logic 0. The $\overline{\text{OE}}$ pin is activated by the 8088 $\overline{\text{RD}}$ signal or the $\overline{\text{MRDC}}$ (**memory read control**) signal of other family members.

If the 20-bit binary address, decoded by the NAND gate, is written so that the leftmost nine bits are 1s and the rightmost 11 bits are don't cares (X), the actual address range of the EPROM can be determined. (A *don't care* is a logic 1 or a logic 0, whichever is appropriate.)

Example 10–1 illustrates how the address range for this EPROM is determined by writing down the externally decoded address bits (A_{19}–A_{11}) and the address bits decoded by the EPROM (A_{10}–A_0) as don't cares. We really do not care about the address pins on the EPROM because they are internally decoded. As the example illustrates, the don't cares are first written as 0s to locate the lowest address and then as 1s to find the highest address. Example 10–1 also shows these binary boundaries as hexadecimal addresses. Here, the 2K EPROM is decoded at memory address locations FF800H–FFFFFH. Notice that this is a 2K-byte section of the memory and is also located at the reset location for the 8086/8088 (FFFF0H), the most likely place for an EPROM in a system.

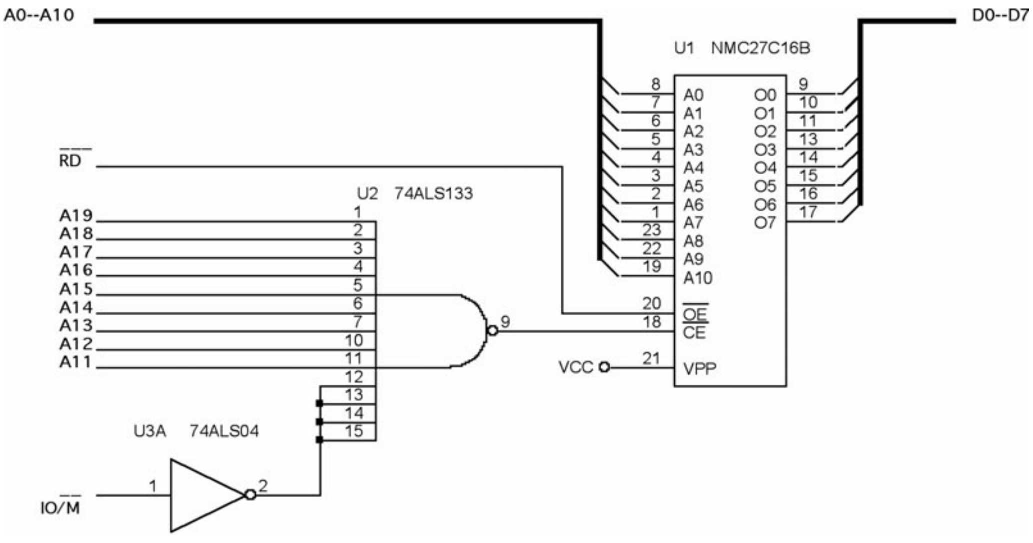


FIGURE 10–13 A simple NAND gate decoder that selects a 2716 EPROM for memory location FF800H–FFFFFH.

EXAMPLE 10–1

```
1111 1111 1XXX XXXX XXXX

          or

1111 1111 1000 0000 0000 = FF800H
          to
1111 1111 1111 1111 1111 = FFFFFH
```

Although this example serves to illustrate decoding, NAND gates are rarely used to decode memory because each memory device requires its own NAND gate decoder. Because of the excessive cost of the NAND gate decoder and inverters that are often required, this option requires that an alternate be found.

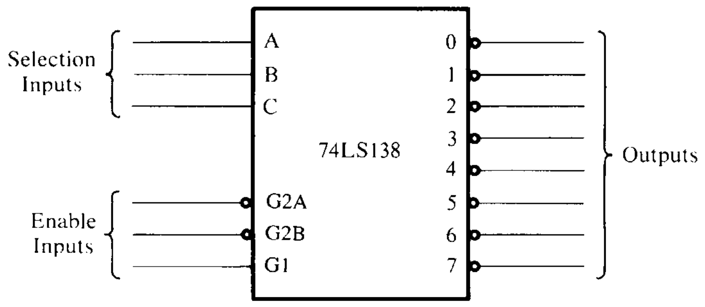
The 3-to-8 Line Decoder (74LS138)

One of the more common, although not only, integrated circuit decoders found in many microprocessor-based systems is the 74LS138 3-to-8 line decoder. Figure 10–14 illustrates this decoder and its truth table.

The truth table shows that only one of the eight outputs ever goes low at any time. For any of the decoder's outputs to go low, the three enable inputs ($\overline{G2A}$, $\overline{G2B}$, and G1) must all be active. To be active, the $\overline{G2A}$ and $\overline{G2B}$ inputs must both be low (logic 0), and G1 must be high (logic 1). Once the 74LS138 is enabled, the address inputs (C, B, and A) select which output pin

Self study

FIGURE 10–14 The 74LS138 3-to-8 line decoder and function table.



Inputs						Outputs							
Enable			Select										
$\overline{G2A}$	$\overline{G2B}$	G1	C	B	A	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$	$\overline{6}$	$\overline{7}$
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0

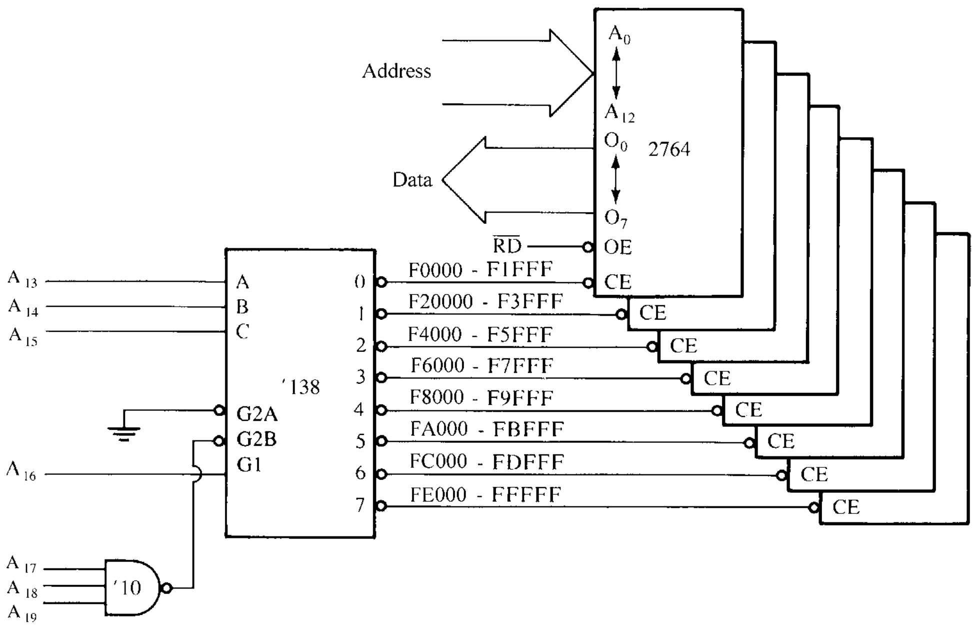


FIGURE 10–15 A circuit that uses eight 2764 EPROMs for a 64K × 8 section of memory in an 8088 microprocessor-based system. The addresses selected in this circuit are F0000H–FFFFFH.

goes low. Imagine eight EPROM $\overline{CE}$ inputs connected to the eight outputs of the decoder! This is a very powerful device because it selects eight different memory devices at the same time. Even today this device still finds wide application.

Sample Decoder Circuit. Notice that the outputs of the decoder, illustrated in Figure 10–15, are connected to eight different 2764 EPROM memory devices. Here, the decoder selects eight 8K-byte blocks of memory for a total memory capacity of 64K bytes. This figure also illustrates the address range of each memory device and the common connections to the memory devices. Notice that all of the address connections from the 8088 are connected to this circuit. Also, notice that the decoder's outputs are connected to the $\overline{CE}$ inputs of the EPROMs, and the $\overline{RD}$ signal from the 8088 is connected to the $\overline{OE}$ inputs of the EPROMs. This allows only the selected EPROM to be enabled and to send its data to the microprocessor through the data bus whenever $\overline{RD}$ becomes a logic 0.

In this circuit, a three-input NAND gate is connected to address bits A_{19}–A_{17}. When all three address inputs are high, the output of this NAND gate goes low and enables input $\overline{G2B}$ of the 74LS138. Input G1 is connected directly to A_{16}. In other words, in order to enable this decoder, the first four address connections (A_{19}–A_{16}) must all be high.

The address inputs C, B, and A connect to microprocessor address pins A_{15}–A_{13}. These three address inputs determine which output pin goes low and which EPROM is selected whenever the 8088 outputs a memory address within this range to the memory system.

Example 10–2 shows how the address range of the entire decoder is determined. Notice that the range is location F0000H–FFFFFH. This is a 64K-byte span of the memory.

EXAMPLE 10–2

```
1111 XXXX XXXX XXXX XXXX

        or

1111 0000 0000 0000 0000 = FOOOOH
        to
1111 1111 1111 1111 1111 = FFFFFH
```

How is it possible to determine the address range of each memory device attached to the decoder's outputs? Again, the binary bit pattern is written down; this time the C, B, and A address inputs are not don't cares. Example 10–3 shows how output 0 of the decoder is made to go low to select the EPROM attached to that pin. Here, C, B, and A are shown as logic 0s.

EXAMPLE 10–3

```
     CBA
1111 OOOX XXXX XXXX XXXX

          or

1111 0000 0000 0000 0000 = FOOOOH
          to
1111 0001 1111 1111 1111 = F1FFFH
```

If the address range of the EPROM connected to output 1 of the decoder is required, it is determined in exactly the same way as that of output 0. The only difference is that now the C, B, and A inputs contain a 001 instead of a 000 (see Example 10–4). The remaining output address ranges are determined in the same manner by substituting the binary address of the output pin into C, B, and A.

EXAMPLE 10–4

```
     CBA
1111 001X XXXX XXXX XXXX

          or

1111 0010 0000 0000 0000 = F2000H
          to
1111 0011 1111 1111 1111 = F3FFFH
```

The Dual 2-to-4 Line Decoder (74LS139)

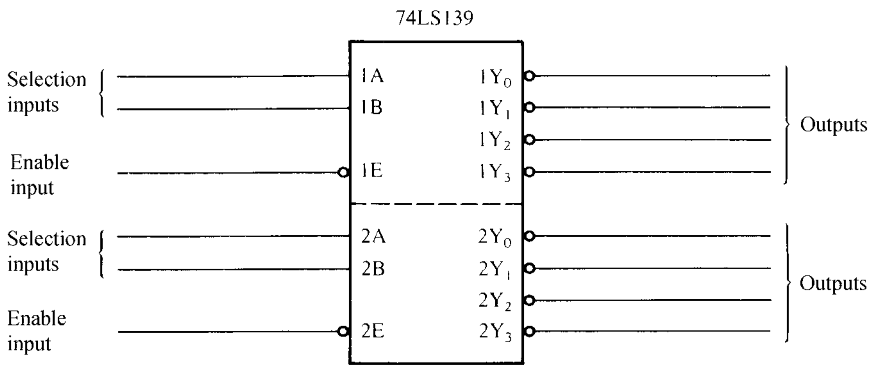
Another decoder that finds some application is the 74LS139 dua1 2-to-4 line decoder. Figure 10–16 illustrates both the pin-out and the truth table for this decoder. The 74LS139 contains two separate 2-to-4 line decoders—each with its own address, enable, and output connections.

A more complicated decoder using the 74LS139 decoder appears in Figure 10–17. This circuit uses a 128K × 8 EPROM (271000) and a 128K × 8 SRAM (621000). The EPROM is decoded at memory locations E0000H–FFFFFH and the SRAM is decoded at addresses 00000H–1FFFFH. This is fairly typical of a small embedded system, where the EPROM is located at the top of the memory space and the SRAM at the bottom.

Output $\overline{Y0}$ of decoder U1A activates the SRAM whenever address bits A_{17} and A_{18} are both logic 0s if the $IO/\overline{M}$ signal is a logic 0 and address line A_{19} is a logic 0. This selects the SRAM for any address between 00000H and 1FFFFH. The second decoder (U1B) is slightly more complicated because the NAND gate (U4B) selects the decoder when $IO/\overline{M}$ is a logic 0 while A_{19} is a logic 1. This selects the EPROM for addresses E0000H through FFFFFH.

PLD Programmable Decoders

This section of the text explains the use of the programmable logic device, or PLD, as a decoder. There are three SPLD (**simple PLD**) devices that function in the same manner but have different names: PLA (**programmable logic array**), PAL (**programmable array logic**), and GAL (**gated array logic**). Although these devices have been in existence since the mid-1970s, they have only appeared in memory system and digital designs since the early 1990s. The PAL and the PLA are fuse-programmed, as is the PROM, and some PLD devices are erasable devices (as are EPROMs). In essence, all three devices are arrays of logic elements that are programmable.



Inputs			Outputs			
$\overline{E}$	A	B	$\overline{Y_0}$	$\overline{Y_1}$	$\overline{Y_2}$	$\overline{Y_3}$
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	X	X	1	1	1	1

FIGURE 10–16 The pin-out and truth table of the 74LS139, dual 2-to-4 line decoder.

Other PLDs are also available, such as CPLDs (**complex programmable logic devices**), FPGAs (**field programmable gate arrays**), and FPICs (**field programmable interconnect**). These types of PLDs are much more complex than the SPLDs that are used more commonly in designing a complete system. If the concentration is on decoding addresses, the SPLD is used and if the concentration is on a complete system, then the CPLD, FPLG, or FPIC is used to implement the design. These devices are also referred to as an ASIC (**application-specific integrated circuit**).

Combinatorial Programmable Logic Arrays. One of the two basic types of PALs is the combinatorial programmable logic array. This device is internally structured as a programmable array of combinational logic circuits. Figure 10–18 illustrates the internal structure of the PAL16L8 that is constructed with AND/OR gate logic. This device, which is representative of a PLD, has 10 fixed inputs, two fixed outputs, and six pins that are programmable as inputs of outputs. Each output signal is generated from a seven-input OR gate that has an AND gate attached to each input. The outputs of the OR gates pass through a three-state inverter that defines each output as an AND/NOR function. Initially, all of the fuses connect all of the vertical/horizontal connections illustrated in Figure 10–18. Programming is accomplished by blowing fuses to connect various inputs to the OR gate array. The wired-AND function is performed at each input connection, which allows a product term of up to 16 inputs. A logic expression using the PAL16L8 can have up to seven product terms with up to 16 inputs NORed together to generate the output expression. This device is ideal as a memory address decoder because of its structure. It is also ideal because the outputs are active low.

Fortunately, we don't have to choose the fuses by number for programming, as was customary when this device was first introduced. A PAL is programmed with a software package such as PALASM, the PAL assembler program. More recently, PLD design is accomplished using HDL (**hardware description language**) or VHDL (**verilog HDL**). The VHDL language and its syntax are currently the industry standard for programming PLD devices. Example 10–5 shows a program that decodes the same areas of memory as decoded in Figure 10–17. Note that this program

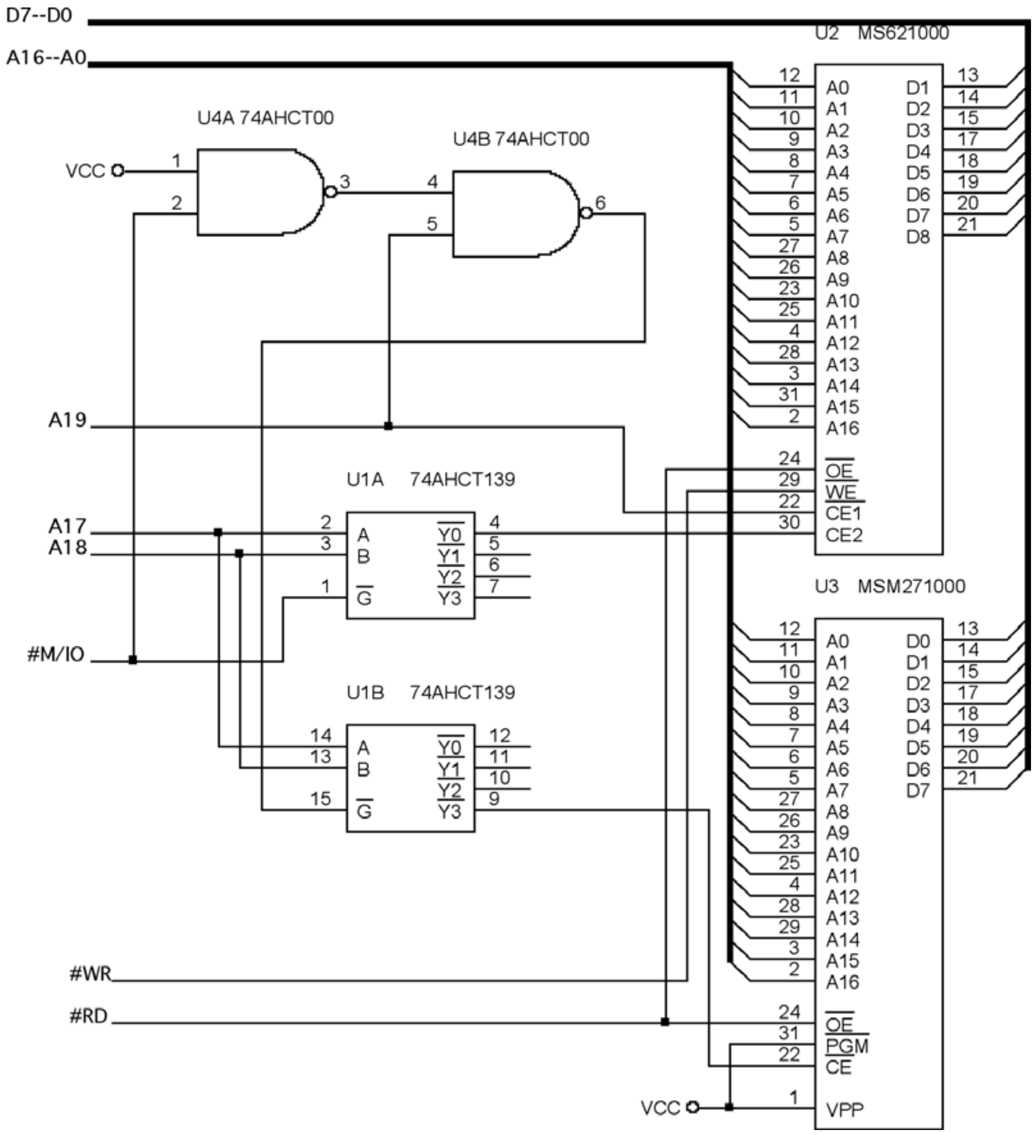


FIGURE 10–17 A sample memory system constructed with a 74HCT139.

was developed by using a text editor such as EDIT, available with Microsoft DOS version 7.1 with XP or NotePad in Windows. The program can also be developed by using an editor than comes with any of the many programming packages for PLDs. Various editors attempt to ease the task of defining the pins, but we believe it is easier to use NotePad and the listing as shown.

EXAMPLE 10–5

```
-- VHDL code for the decoder of Figure 10–17

library ieee;
use ieee.std_logic_1164.all;

entity DECODER_10_19 is

port (
       A19, A18, A17, MIO: in STD_LOGIC;
       ROM, RAM, AX19: out STD_LOGIC
```

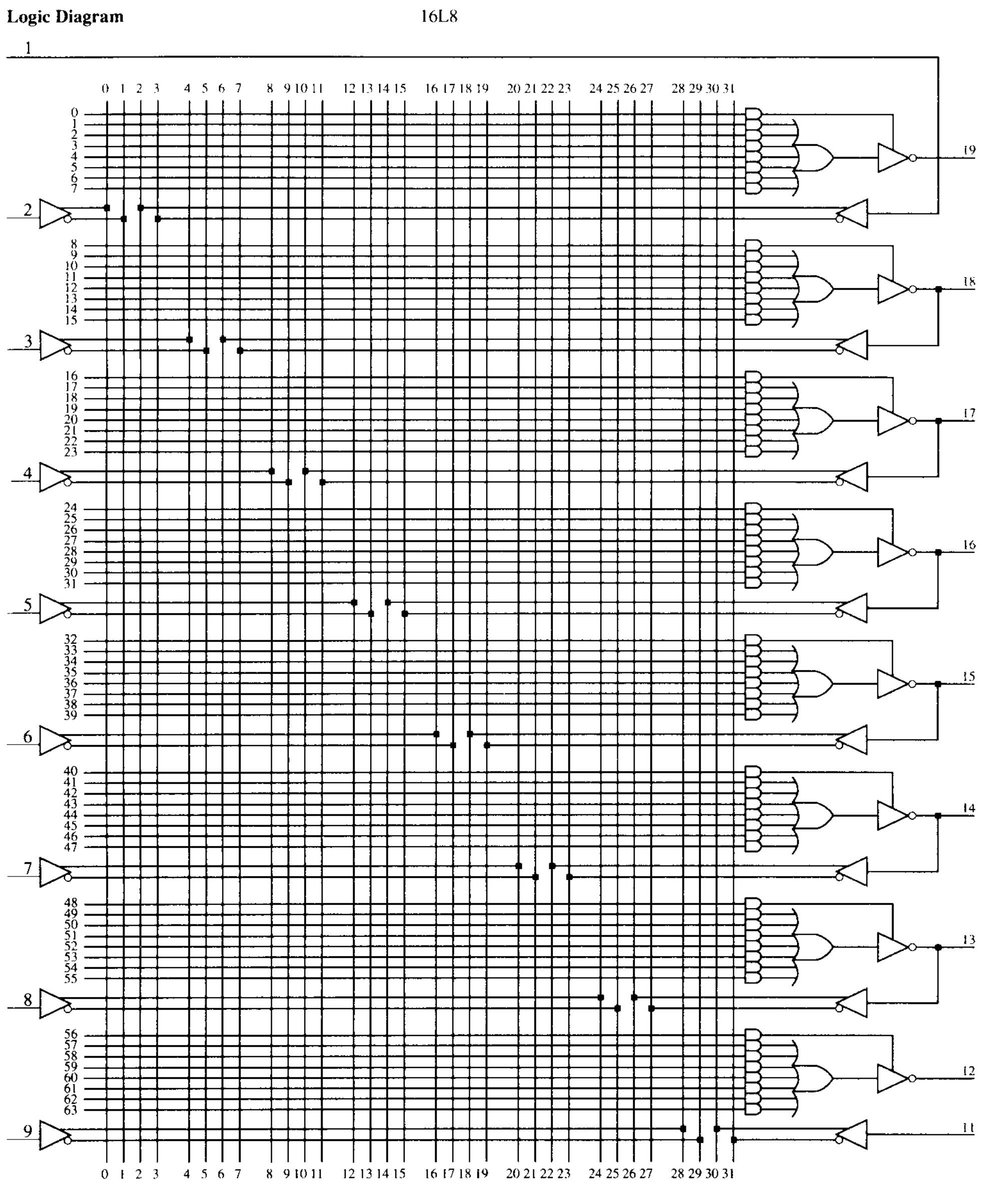


FIGURE 10–18 The PAL16L8. (Copyright Advanced Micro Devices, Inc., 1988. Reprinted with permission of copyright owner. All rights reserved.)

```
);

end;

architecture V1 of DECODER_10_19 is

begin

       ROM <= A19 or A18 or A17 or MIO;
       RAM <= not (A18 and A17 and (not MIO));
       AX19 <= not A19;

end V1;
```

Comments in VHDL programming begin with a pair of minus signs as illustrated in the first line of the VHDL code in Example 10–5. The library and use statements specify the standard IEEE library using standard logic. The entity statement names the VHDL module, in this case DECODER_10_17. The port statements define the in, out, and in-out pins used in the equations for the logic expression, which appears in the begin block. A_{19}, A_{18}, A_{17}, and MIO (this signal cannot be defined as IO/$\overline{\text{M}}$ so it was called MIO) are defined as input pins and ROM and RAM are the output pins for connection to the $\overline{\text{CS}}$ pins on the memory devices. The architecture statement merely refers to the version (V_1) of this design. Finally, the equations for the design are placed in the begin block. Each output pin has its own equation. The keyword not is used for logical inversion and the keyword and is used for the logical and operation. In this case the ROM equation causes the ROM pin to become a logic zero only when the A_{19}, A_{18}, A_{17}, and MIO are all logic zeros (00000H–1FFFFH). The RAM equation causes the RAM pin to become a logic zero when A_{18} and A_{17} are all ones at the same time that MIO is a logic zero. A_{19} is connected to the active high CE2 pin after being inverted by the PLD. The RAM is selected for addresses 60000H–7FFFFH. See Figure 10–19 for the PLD realization of Example 10–5.

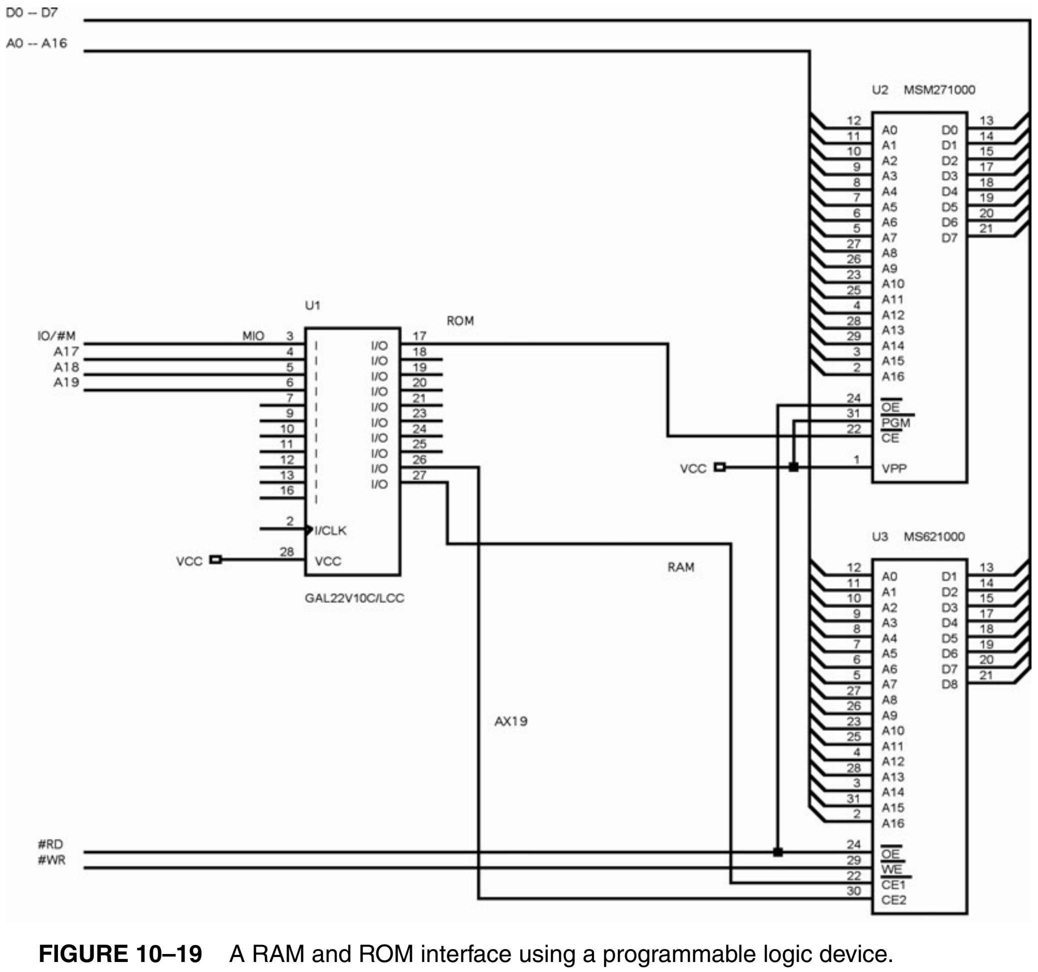


FIGURE 10–19 A RAM and ROM interface using a programmable logic device.

10–3 8088 AND 80188 (8-BIT) MEMORY INTERFACE

This text contains separate sections on memory interfacing for the 8088 and 80188 with their 8-bit data buses; the 8086, 80186, 80286, and 80386SX with their 16-bit data buses; the 80386DX and 80486 with their 32-bit data buses; the Pentium–Core2 with their 64-bit data buses. Separate sections are provided because the methods used to address the memory are slightly different in microprocessors that contain different data bus widths. Hardware engineers or technicians who wish to broaden their expertise in interfacing 16-bit, 32-bit, and 64-bit memory interface should cover all sections. This section is much more complete than the sections on the 16-, 32-, and 64-bit-wide memory interface, which cover material not explained in the 8088/80188 section.

In this section, we examine the memory interface to both RAM and ROM and explain the error-correction code (ECC), which is still is currently available to memory system designers. Many home computer systems do not use ECC because of the cost, but business machines often do use it.

Basic 8088/80188 Memory Interface

The 8088 and 80188 microprocessors have an 8-bit data bus, which makes them ideal to connect to the common 8-bit memory devices available today. The 8-bit memory size makes the 8088, and especially the 80188, ideal as a simple controller. For the 8088/80188 to function correctly with the memory, however, the memory system must decode the address to select a memory component. It must also use the $\overline{RD}$, $\overline{WR}$, and IO/$\overline{M}$ control signals provided by the 8088/80188 to control the memory system.

The minimum mode configuration is used in this section and is essentially the same as the maximum mode system for memory interface. The main difference is that, in maximum mode, the IO/$\overline{M}$ signal is combined with $\overline{RD}$ to generate the $\overline{MRDC}$ signal, and IO/$\overline{M}$ is combined with $\overline{WR}$ to generate the $\overline{MWTC}$ signal. The maximum mode control signals are developed inside the 8288 bus controller. In minimum mode, the memory sees the 8088 or the 80188 as a device with 20 address connections (A_{19}–A_0), eight data bus connections (AD_7–AD_0), and the control signals IO/$\overline{M}$, $\overline{RD}$, and $\overline{WR}$.

Interfacing EPROM to the 8088. You will find this section very similar to Section 10–2 on decoders. The only difference is that, in this section, we discuss wait states and the use of the IO/$\overline{M}$ signal to enable the decoder.

Figure 10–20 illustrates an 8088/80188 microprocessor connected to three 27256 EPROMs, $32K \times 8$ memory devices. The 27256 has one more address input (A_{15}) than the 27128 and twice the memory. The 74HCT138 decoder in this illustration decodes three $32K \times 8$ blocks of memory for a total of $96K \times 8$ bits of the physical address space for the 8088/80188.

The decoder (74HCT138) is connected a little differently than might be expected because the slower version of this type of EPROM has a memory access time of 450 ns. Recall from Chapter 9 that when the 8088 is operated with a 5 MHz clock, it allows 460 ns for the memory to access data. Because of the decoder's added time delay (8 ns), it is impossible for this memory to function within 460 ns. In order to correct this problem, the output from the NAND gate can be used to generate a signal to enable the decoder and a signal for the wait state generator, covered in Chapter 9. (Note that the 80188 can internally insert from 0 to 15 wait states without any additional external hardware, so it does not require this NAND gate.) With a wait state inserted every time this section of the memory is accessed, the 8088 will allow 660 ns for the EPROM to access data. Recall that an extra wait state adds 200 ns (1 clock) to the access time. The 660 ns is ample time for a 450 ns memory component to access data, even with the delays introduced by the decoder and any buffers added to the data bus. The wait states are inserted in this system for memory locations C0000H–FFFFFH. If this creates a problem, a three-input OR gate can be added to the three outputs of the decoder to generate a wait signal only for the actual addresses for this system (E8000H–FFFFFH).

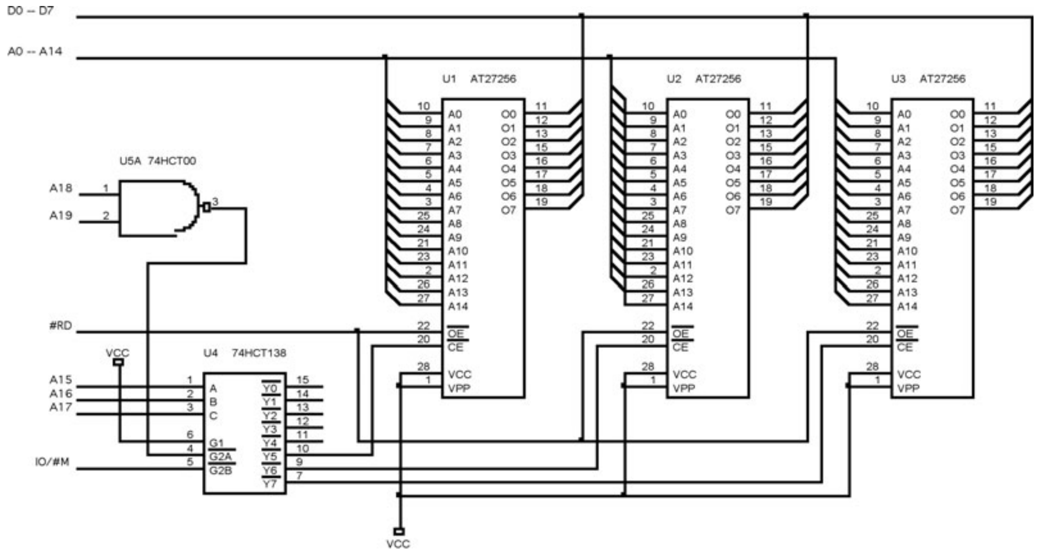


FIGURE 10–20 Three 27256 EPROMs interfaced to the 8088 microprocessor.

Notice that the decoder is selected for a memory address range that begins at location E8000H and continues through location FFFFFH—the upper 96K bytes of memory. This section of memory is an EPROM because FFFF0H is where the 8088 starts to execute instructions after a hardware reset. We often call location FFFF0H the **cold-start location**. The software stored in this section of memory would contain a JMP instruction at location FFFF0H that jumps to location E8000H so the remainder of the program can execute. In this circuit, U_1 is decoded at addresses E8000H–EFFFFH, U_2 is decoded at F0000H–F7FFFH, and U_3 is decoded at F8000H–FFFFFH.

Interfacing RAM to the 8088. RAM is a little easier to interface than EPROM because most RAM memory components do not require wait states. An ideal section of the memory for the RAM is the very bottom, which contains vectors for interrupts. Interrupt vectors (discussed in more detail in Chapter 12) are often modified by software packages, so it is rather important to encode this section of the memory with RAM.

Figure 10–21 shows sixteen 62256, 32K × 8 static RAMs interfaced to the 8088, beginning at memory location 00000H. This circuit board uses two decoders to select the 16 different RAM memory components and a third to select the other decoders for the appropriate memory sections. Sixteen 32K RAMs fill memory from location 00000H through location 7FFFFH, for 512K bytes of memory.

The first decoder (U_4) in this circuit selects the other two decoders. An address beginning with 00 selects decoder U_3 and an address that begins with 01 selects decoder U_9. Notice that extra pins remain at the output of decoder U_4 for future expansion. These pins allow more 256K × 8 blocks of RAM for a total of 1M × 8, simply by adding the RAM and the additional secondary decoders.

Also notice from the circuit in Figure 10–21 that all the address inputs to this section of memory are buffered, as are the data bus connections and control signals $\overline{RD}$ and $\overline{WR}$. Buffering is important when many devices appear on a single board or in a single system. Suppose that three other boards like this are plugged into a system. Without the buffers on each board, the load on the system address, data, and control buses would be enough to prevent proper operation. (Excessive loading causes the logic 0 output to rise above the 0.8 V maximum allowed in a system.) Buffers are normally used if the memory will contain additions at some future date. If the memory will never grow, then buffers may not be needed.

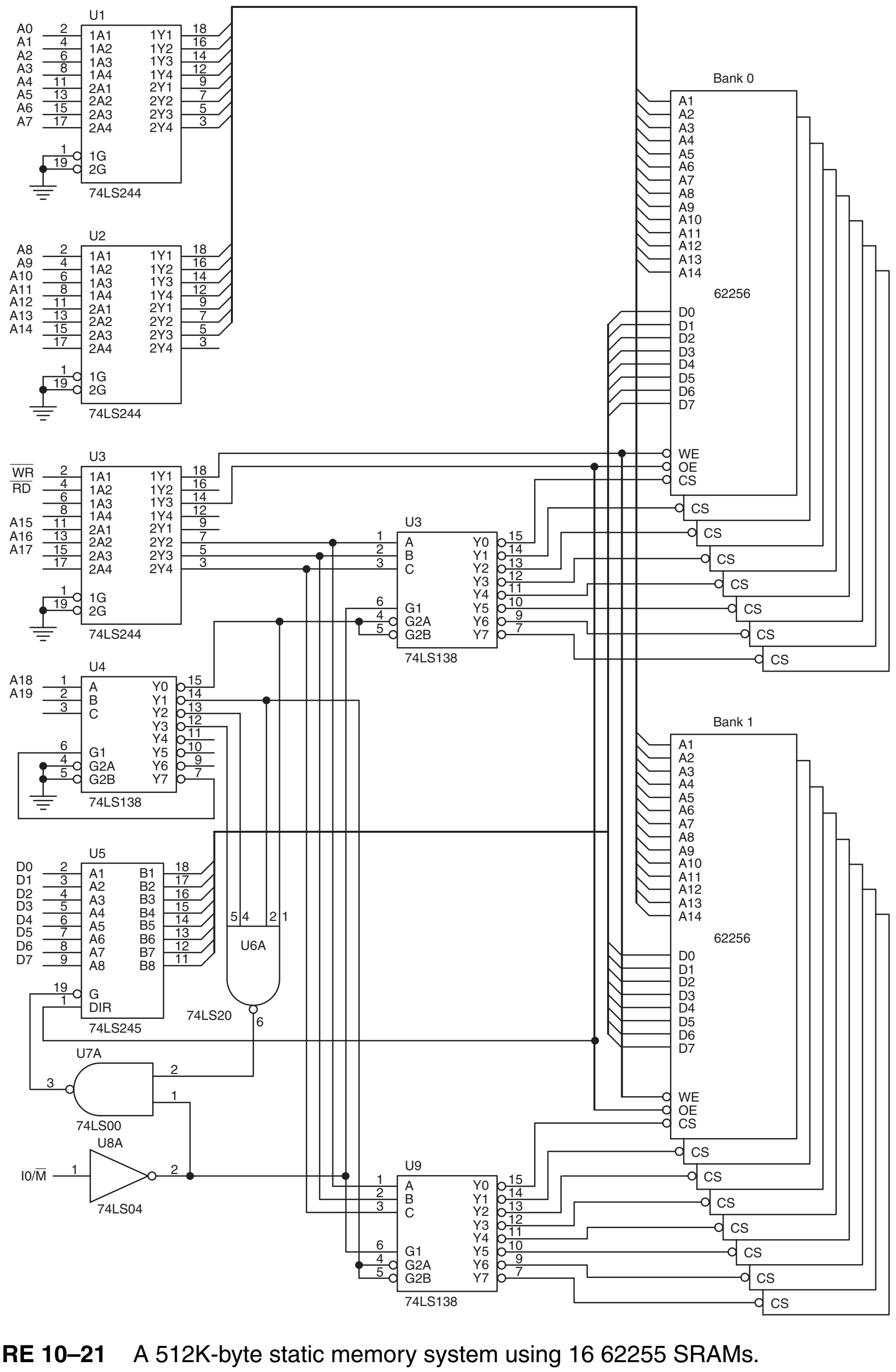


FIGURE 10–21 A 512K-byte static memory system using 16 62255 SRAMs.

Interfacing Flash Memory

Flash memory (EEPROM) is becoming commonplace for storing setup information on video cards, as well as for storing the system BIOS in the personal computer. It even finds application in MP3 audio players and USB pen drives. Flash memory is also found in many other applications to store information that is only changed occasionally.

The only difference between a flash memory device and SRAM is that the flash memory device requires a 12V programming voltage to erase and write new data. The 12V can be available either at the power supply or a 5V to 12V converter designed for use with flash memory can be obtained. The newest versions of flash memory are erased with a 5.0V or even a 3.3V signal so that a converter is not needed.

EEPROM is available as either a memory device with a parallel interface or as devices that are serial. The serial device is extremely small and is not suited for memory expansion, but as an I/O device it can store information such as in a flash drive. This section of the text details both memory types.

Figure 10–22 illustrates the 28F400 Intel flash memory device interfaced to the 8088 microprocessor using its parallel interface. The 28F400 can be used as either a 512K × 8 memory device or as a 256K × 16 memory device. Because it is interfaced to the 8088, its configuration is 512K × 8. Notice that the control connections on this device are identical to that of an SRAM: $\overline{CE}$, $\overline{OE}$, and $\overline{WE}$. The only new pins are V_{PP}, which is connected to 12V for erase and programming; $\overline{PWD}$, which selects the power-down mode when a logic 0 and is also used for programming; and $\overline{BYTE}$, which selects byte (0) or word (1) operation. Note that the pin DQ_{15} functions as the least significant address input when operated in the byte mode. Another difference is the amount of time required to accomplish a write operation. The SRAM can perform a write operation in as little as 10 ns, but the flash memory requires approximately 0.4 seconds to erase a byte. The topic of programming the flash memory device is covered in Chapter 11, along with I/O devices. The flash memory device has internal registers that are programmed by using I/O techniques not yet explained. This chapter concentrates on its interface to the microprocessor.

Notice in Figure 10–22 that the decoder chosen is the 74LS139 because only a simple decoder is needed for a flash memory device this large. The decoder uses address connection A_{19} and IO/$\overline{M}$ as inputs. The A_{15} signal selects the flash memory for locations 80000H through FFFFFH, and IO/$\overline{M}$ enables the decoder.

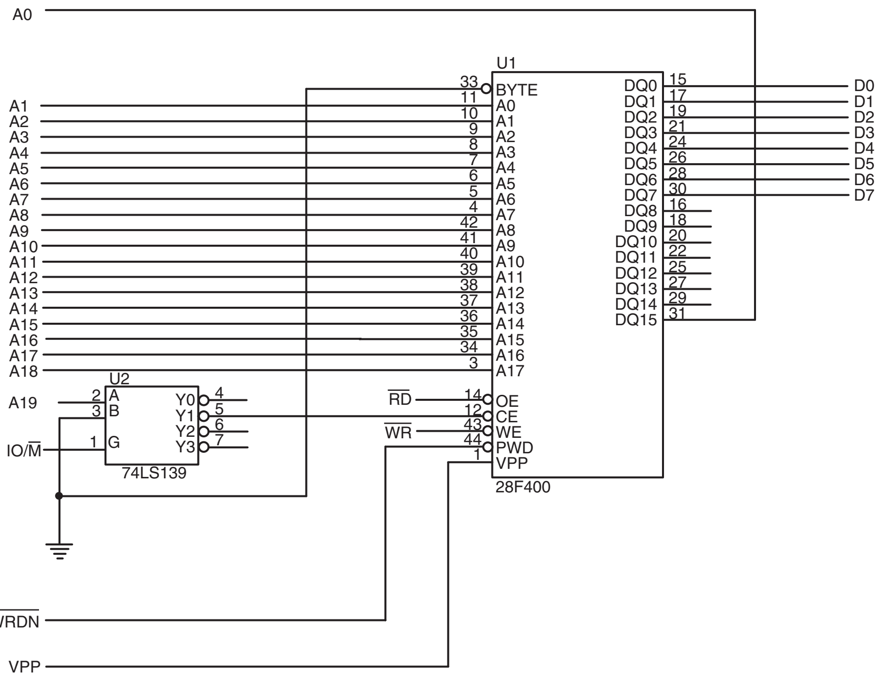


FIGURE 10–22 The 28F400 flash memory device interfaced to the 8088 microprocessor.

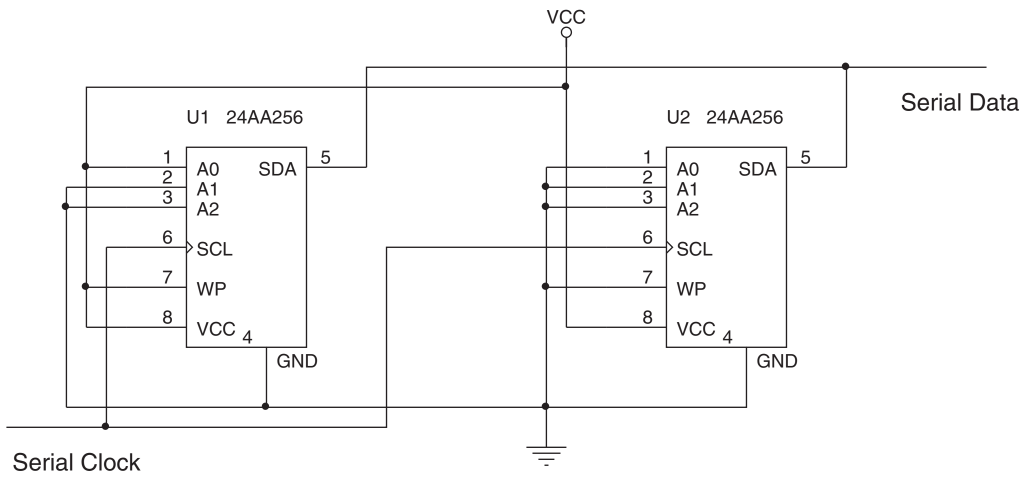


FIGURE 10–23 A serial EEPROM interface.

As mentioned, many newer flash memory devices use a serial interface to reduce the cost of the integrated circuit because of fewer pins and a smaller size. Serial flash memory is available in sizes to 4G bytes and has comparable speeds and erase times with the parallel flash devices. Most modern flash memory functions from 5V or 3.3V without the need for a higher programming voltage and has a life of 1,000,000 erases with a storage time of 200 years.

Figure 10–23 illustrates a small serial flash device (a 256K device, organized as a 32K × 8 memory). The three address pins are hardwired to allow more than one device to be placed on a serial bus. In the illustration U_1 is wired at address 001 and U_2 is wired at address 000. Not shown in the illustration is a pull-up resistor that is needed for the serial data connection. The pull-up may be located in the microprocessor or it may need to be connected externally, depending on the microprocessor and interface connected to the memory.

This memory interface has two signal lines. One is a serial clock (SCL) and the other is a bidirectional serial data line (SDA). The clock frequency can be anything up to 400 KHz, so this type of memory is not meant to replace the main memory in a system. It is fast enough for music or other low-speed data. The serial interface is explained in Chapter 11.

Figure 10–24 depicts the basic serial data format for the serial EEPROM. The serial data contains the address (the A_0, A_1, A_2 pins) in the first byte as well as a device code of 1010, which represents the EEPROM. Other serial devices have different device codes. This is followed by the memory location and the data in additional bytes.

Error Correction

Error-correction schemes have been around for a long time, but integrated circuit manufacturers have only recently started to produce error-correcting circuits. One such circuit is the 74LS636, an 8-bit error correction and detection circuit that corrects any single-bit memory read error and flags any 2-bit error called SECDED (**single error correction/double error correction**). This device is found in high-end computer systems because of the cost of implementing a system that uses error correction.

The newest computer systems are now using DDR memory with ECC (error-correction code). The scheme to correct the errors that might occur in these memory devices is identical to the scheme discussed in this text.

Write Address

S	1	0	1	0	A2	A1	A0	0	ACK	x	A14	A13	A12	A11	A10	A9	A8	ACK	A7	A6	A5	A4	A3	A2	A1	A0	ACK

Followed by for a Read Byte

S	1	0	1	0	A2	A1	A0	1	ACK	D7	D6	D5	D4	D3	D2	D1	D0	ACK	P

or Followed by for a Write Byte

D7	D6	D5	D4	D3	D2	D1	D0	ACK	P

S = Start
P = Stop
ACK = acknowledge

FIGURE 10–24 The data signals to the serial EEPROM for a read or a write.

The 74LS636 corrects errors by storing five parity bits with each byte of memory data. This does increase the amount of memory required, but it also provides automatic error correction for single-bit errors. If more than two bits are in error, this circuit may not detect it. Fortunately, this is rare, and the extra effort required to correct more than a single-bit error is very expensive and not worth the effort. Whenever a memory component fails completely, its bits are all high or all low. In this case, the circuit flags the processor with a multiple-bit error indication.

Figure 10–25 depicts the pin-out of the 74LS636. Notice that it has eight data I/O pins, five check bit I/O pins, two control inputs (SO and SI), and two error outputs: single-error flag (SEF) and double-error flag (DEF). The control inputs select the type of operation to be performed and are listed in the truth table of Table 10–1.

When a single error is detected, the 74LS636 goes through an error-correction cycle. It places a 01 on S_0 and S_1 by causing a wait and then a read following error correction.

Figure 10–26 illustrates a circuit used to correct single-bit errors with the 74LS636 and to interrupt the processor through the NMI pin for double-bit errors. To simplify the illustration, we depict only one 2K × 8 RAM and a second 2K × 8 RAM to store the 5-bit check code.

The connection of this memory component is different from that of the previous example. Notice that the S or $\overline{\text{CS}}$ pin is grounded, and data bus buffers control the flow to the system bus. This is necessary if the data are to be accessed from the memory before the $\overline{\text{RD}}$ strobe goes low.

On the next negative edge of the clock after the $\overline{\text{RD}}$ signal, the 74LS636 checks the single-error flag (SEF) to determine whether an error has occurred. If it has, a correction cycle causes the single-error defect to be corrected. If a double error occurs, an interrupt request is generated by the double-error flag (DEF) output, which is connected to the NMI pin of the microprocessor.

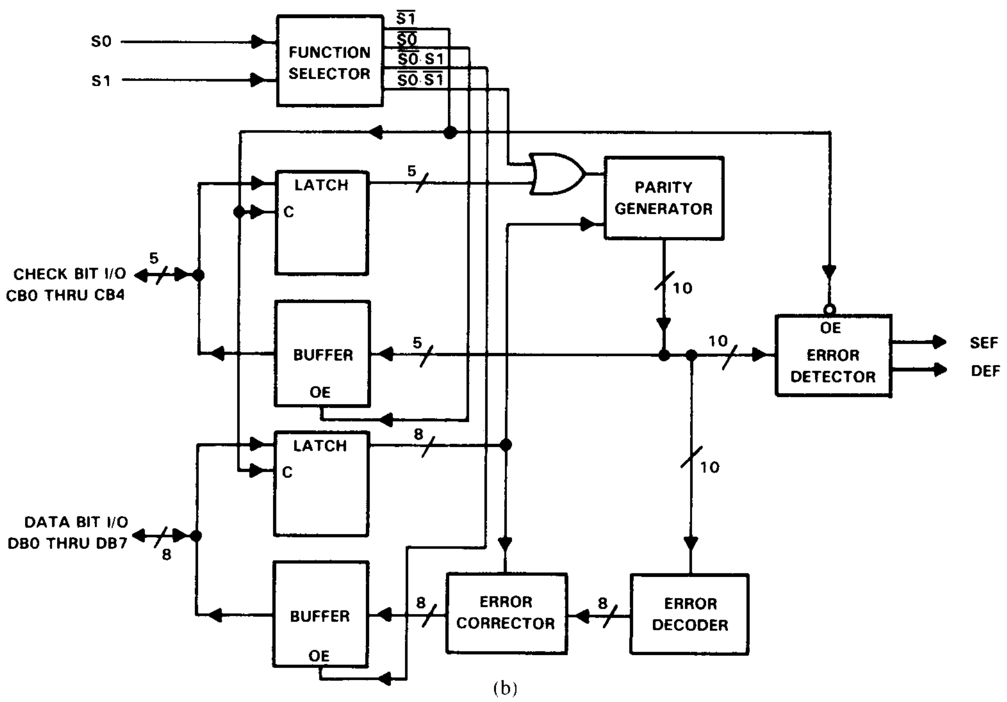
Modern DDR error-correction memory (ECC) does not actually have logic circuitry on board that detects and corrects errors. Since the Pentium, the microprocessor incorporates the logic circuitry to detect/correct errors provided the memory can store the extra 8 bits required for storing the ECC code. ECC memory is 72-bits wide using the additional 8 bits to store the ECC code. If an error occurs, the microprocessor runs the correction cycle to correct the error. Some memory devices such as Samsung memory also perform an internal error check. The Samsung ECC uses 3 bytes to check every 256 bytes of memory, which is far more efficient. Additional information on the Samsung ECC algorithm is available at the Samsung website.

pin assignments

J, N PACKAGES			
1	DEF	11	CB4
2	DB0	12	nc
3	DB1	13	CB3
4	DB2	14	CB2
5	DB3	15	CB1
6	DB4	16	CB0
7	DB5	17	S0
8	DB6	18	S1
9	DB7	19	SEF
10	GND	20	V_{CC}

(a)

functional block diagram



(b)

FIGURE 10–25 (a) The pin connections of the 74LS636. (b) The block diagram of the 74LS636. (Courtesy of Texas Instruments Incorporated.)

TABLE 10–1 Control bits S_0 and S_1 for the 74LS636.

S_1	S_0	Function	SEF	DEF
0	0	Write check word	0	0
0	1	Correct data word	*	*
1	0	Read data	0	0
1	1	Latch data	*	*

*Note: These levels are determined by the error type.

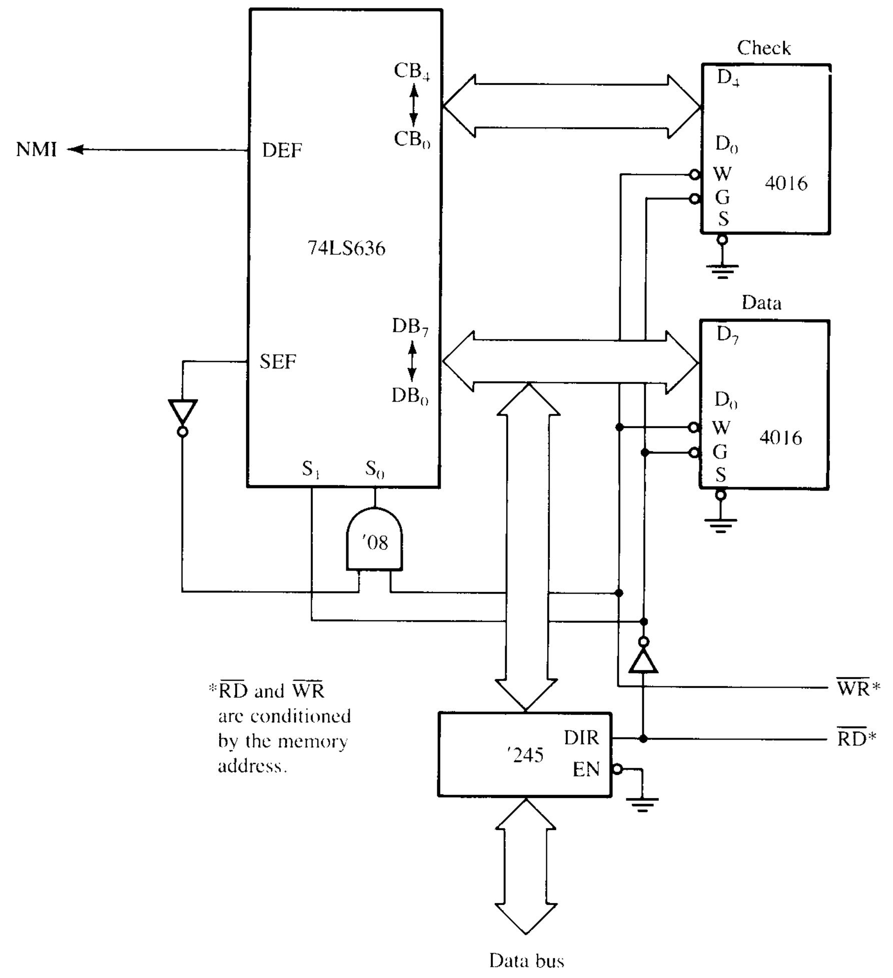


FIGURE 10–26 An error detection and correction circuit using the 74LS636.

10–4 8086, 80186, 80286, AND 80386SX (16-BIT) MEMORY INTERFACE

The 8086, 80186, 80286, and 80386SX microprocessors differ from the 8088/80188 in three ways: (1) The data bus is 16 bits wide instead of 8 bits wide as on the 8088; (2) the $IO/\overline{M}$ pin of the 8088 is replaced with an $M/\overline{IO}$ pin; and (3) there is a new control signal called bus high enable ($\overline{BHE}$). The address bit A_0 or $\overline{BLE}$ is also used differently. (Because this section is based on information provided in Section 10–3, it is extremely important that you read the previous section first.) A few other differences exist between the 8086/80186 and the 80286/80386SX. The 80286/80386SX contains a 24-bit address bus (A_{23}–A_0) instead of the 20-bit address bus (A_{19}–A_0) of the 8086/80186. The 8086/80186 contain an $M/\overline{IO}$ signal, while the 80286 system and 80386SX microprocessor contain control signals $\overline{MRDC}$ and $\overline{MWTC}$ instead of $\overline{RD}$ and $\overline{WR}$.

16-Bit Bus Control

The data bus of the 8086, 80186, 80286, and 80386SX is twice as wide as the bus for the 8088/80188. This wider data bus presents us with a unique set of problems. The 8086, 80186, 80286, and 80386SX must be able to write data to any 16-bit location—or any 8-bit location.

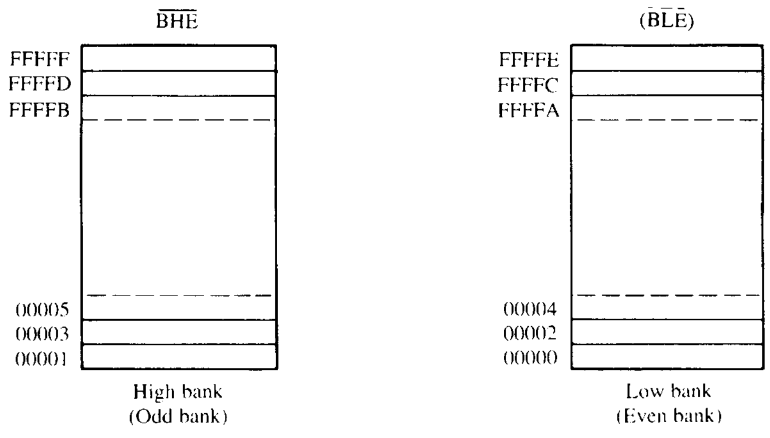


FIGURE 10–27 The high (odd) and low (even) 8-bit memory banks of the 8086/80286/80386SX microprocessors.

This means that the 16-bit data bus must be divided into two separate sections (or **banks**) that are 8 bits wide so that the microprocessor can write to either half (8-bit) or both halves (16-bit). Figure 10–27 illustrates the two banks of the memory. One bank (**low bank**) holds all the even-numbered memory locations, and the other bank (**high bank**) holds all the odd-numbered memory locations.

The 8086, 80186, 80286, and 80386SX use the $\overline{BHE}$ signal (high bank) and the A_0 address bit or $\overline{BLE}$ (bus low enable) to select one or both banks of memory used for the data transfer. Table 10–2 depicts the logic levels on these two pins and the bank or banks selected.

Bank selection is accomplished in two ways: (1) A separate write signal is developed to select a write to each bank of the memory, or (2) separate decoders are used for each bank. As a careful comparison reveals, the first technique is by far the least costly approach to memory interface for the 8086, 80186, 80286, and 80386SX microprocessors. The second technique is only used in a system that must achieve the most efficient use of the power supply.

Separate Bank Decoders. The use of separate bank decoders is often the least effective way to decode memory addresses for the 8086, 80186, 80286, and 80386SX microprocessors. This method is sometimes used, but it is difficult to understand why in most cases. One reason may be to conserve energy, because only the bank or banks selected are enabled. This is not always the case, as with the separate bank read and write signals that are discussed later.

Figure 10–28 illustrates two 74LS138 decoders used to select 64K RAM memory components for the 80386SX microprocessor (24-bit address). Here, decoder U_2 has the $\overline{BLE}$ (A_0) attached to $\overline{G2A}$, and decoder U_3 has the $\overline{BHE}$ signal attached to its $\overline{G2A}$ input. Because the decoder will not activate until all of its enable inputs are active, decoder U_2 activates only for a 16-bit operation or an 8-bit operation from the low bank. Decoder U_3 activates for a 16-bit operation

TABLE 10–2 Memory bank selection using $\overline{BHE}$ and $\overline{BLE}$ (A_0).

$\overline{BHE}$	$\overline{BLE}$	*Function*
0	0	Both banks enabled for a 16-bit transfer
0	1	High bank enabled for an 8-bit transfer
1	0	Low bank enabled for an 8-bit transfer
1	1	No bank enabled

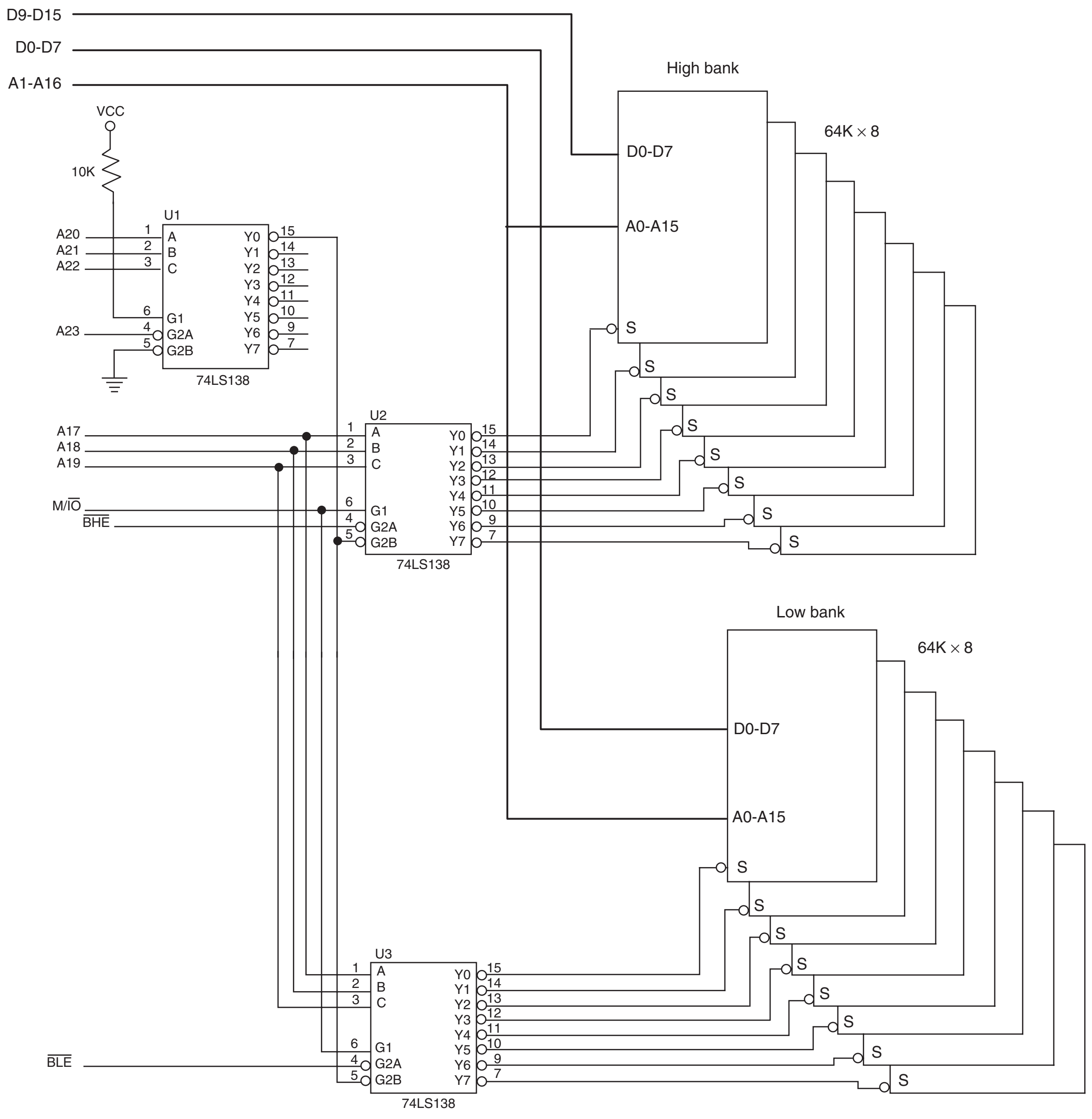


FIGURE 10–28 Separate bank decoders.

or an 8-bit operation to the high bank. These two decoders and the sixteen 64K-byte RAMs they control represent a 1M range of the 80386SX memory system. Decoder U_1 enables U_2 and U_3 for memory address range 000000H–0FFFFFH.

Notice in Figure 10–28 that the A_0 address pin does not connect to the memory because it does not exist on the 80386SX microprocessor. Also notice that address bus bit position A_1 is connected to memory address input A_0, A_2 is connected to A_1, and so forth. The reason is that A_0 from the 8086/80186 (or $\overline{BLE}$ from the 80286/80386SX) is already connected to decoder U_2 and does not need to be connected again to the memory. If A_0 or $\overline{BLE}$ is attached to the A_0 address pin of memory, every other memory location in each bank of memory would be used. This means that half of the memory is wasted if A_0 or $\overline{BLE}$ is connected to A_0.

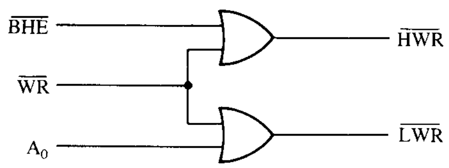


FIGURE 10–29 The memory bank write selection input signals: $\overline{HWR}$ (high bank write) and $\overline{LWR}$ (low bank write).

Separate Bank Write Strobes. The most effective way to handle bank selection is to develop a separate write strobe for each memory bank. This technique requires only one decoder to select a 16-bit-wide memory, which often saves money and reduces the number of components in a system.

Why not also generate separate read strobes for each memory bank? This is usually unnecessary because the 8086, 80186, 80286, and 80386SX microprocessors read only the byte of data that they need at any given time from half of the data bus. If 16-bit sections of data are always presented to the data bus during a read, the microprocessor ignores the 8-bit section that it doesn't need, without any conflicts or special problems.

Figure 10–29 depicts the generation of separate 8086 write strobes for the memory. Here, a 74LS32 OR gate combines A_0 with $\overline{WR}$ for the low bank selection signal ($\overline{LWR}$), and $\overline{BHE}$ combines with $\overline{WR}$ for the high bank selection signal ($\overline{HWR}$). Write strobes, for the 80286/80386SX, are generated by using the $\overline{MWTC}$ signal instead of $\overline{WR}$.

A memory system that uses separate write strobes is constructed differently from either the 8-bit system (8088) or the system using separate memory banks. Memory in a system that uses separate write strobes is decoded as 16-bit-wide memory. For example, suppose that a memory system will contain 64K bytes of SRAM memory. This memory requires two 32K-byte memory devices (62256) so that a 16-bit-wide memory can be constructed. Because the memory is 16 bits wide and another circuit generates the bank write signals, address bit A_0 becomes a don't care. In fact, A_0 is not even a pin on the 80386SX microprocessor.

Example 10–6 shows how a 16-bit-wide memory stored at locations 060000H–06FFFFH is decoded for the 80286 or 80386 microprocessor. Memory in this example is decoded, so bit A_0 is a don't care for the decoder. Bit positions A_1–A_{15} are connected to memory component address pins A_0–A_{14}. The decoder (GAL22V10) enables both memory devices by using address connection A_{23}–A_{15} to select memory whenever address 06XXXXH appears on the address bus.

EXAMPLE 10–6

```
0000 0110 0000 0000 0000 0000 = 060000H
              to
0000 0110 1111 1111 1111 1111 = 06FFFFH

0000 0110 XXXX XXXX xxxx XXXX = 06XXXXH
```

Figure 10–30 illustrates this simple circuit by using a GAL22V10 to both decode memory and generate the separate write strobe. The program for the GAL22V10 decoder is illustrated in Example 10–7. Notice that not only is the memory selected, but both the lower and upper write strobes are also generated by the PLD.

EXAMPLE 10–7

```
-- VHDL code for the decoder of Figure 10-30

library ieee;
use ieee.std_logic_1164.all;

entity DECODER_10_30 is

port (
       A23, A22, A21, A20, A19, A18, A17, A16, A0, BHE, MWTC: in STD_LOGIC;
       SEL, LWR, HWR: out STD_LOGIC
);

end;
```

```
architecture V1 of DECODER_10_30 is
begin
       SEL <= A23 or A22 or A21 or A20 or A19 or (not A18) or (not A17) or A16;
       LWR <= A0 or MWTC;
       HWR <= BHE or MWTC;
end V1;
```

Figure 10–31 depicts a small memory system for the 8086 microprocessor that contains an EPROM section and a RAM section. Here, there are four 27128 EPROMs (16K × 8) that compose a 32K × 16-bit memory at locations F0000–FFFFFH and four 62256 (32K × 8) RAMs that compose 64K × 16-bit memory at locations 00000H–1FFFFH. (Remember that even though the memory is 16 bits wide, it is still numbered in bytes.)

This circuit uses a 74HC139 dual 2-to-4 line decoder that selects EPROM with one half and RAM with the other half. It decodes memory that is 16 bits wide, not 8 bits, as before. Notice that the $\overline{\text{RD}}$ strobe is connected to all the EPROM $\overline{\text{OE}}$ inputs and all RAM $\overline{\text{OE}}$ input pins. This is done because even if the 8086 is reading only 8 bits of data, the application of the remaining 8 bits to the data bus has no effect on the operation of the 8086.

The $\overline{\text{LWR}}$ and $\overline{\text{HWR}}$ strobes are connected to different banks of the RAM memory. Here, it does matter whether the microprocessor is doing a 16-bit or an 8-bit write. If the 8086 writes a 16-bit number to memory, both $\overline{\text{LWR}}$ and $\overline{\text{HWR}}$ go low and enable the $\overline{\text{WE}}$ pins in both memory banks. But if the 8086 does an 8-bit write, only one of the write strobes goes low, writing to only one memory bank. Again, the only time that the banks make a difference is for a memory write operation.

Notice that an EPROM decoder signal is sent to the 8086 wait state generator because EPROM memory usually requires a wait state. The signal comes from the NAND gate used to select the EPROM decoder section, so that if EPROM is selected, a wait state is requested.

Figure 10–32 illustrates a memory system connected to the 80386SX microprocessor by using a GAL22V10 as a decoder. This interface contains 256K bytes of EPROM in the form of

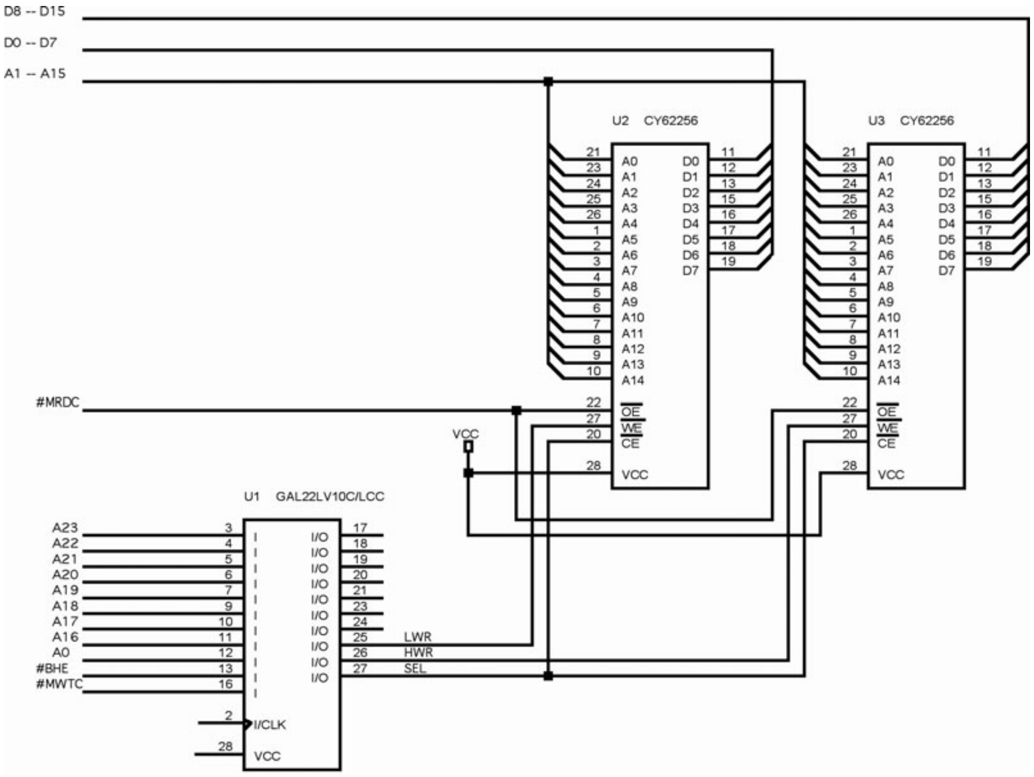


FIGURE 10–30 A 16-bit-wide memory interfaced at memory locations 06000H–06FFFH.

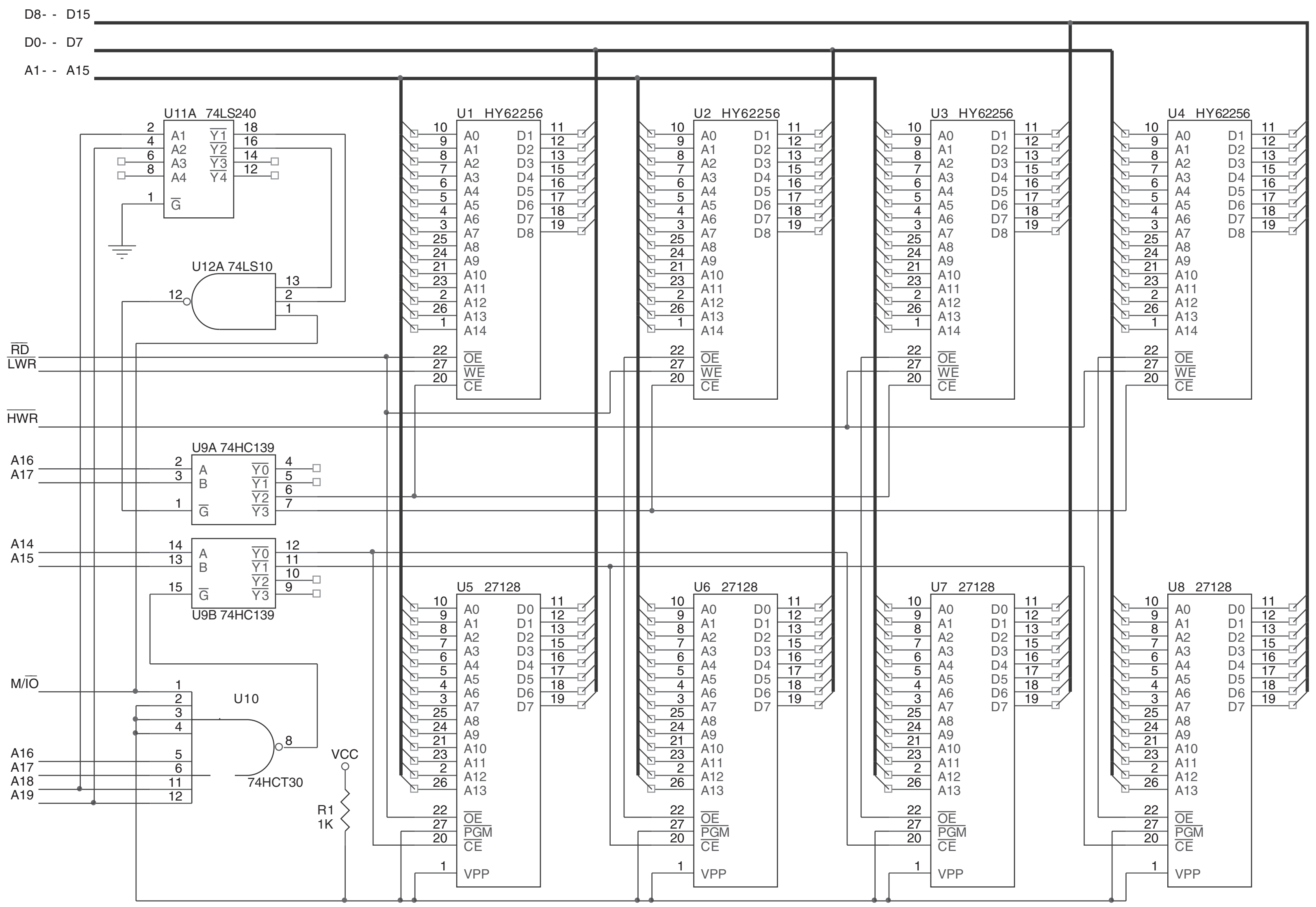


FIGURE 10–31 A memory system for the 8086 that contains a 64K-byte EPROM and a 128K-byte SRAM.

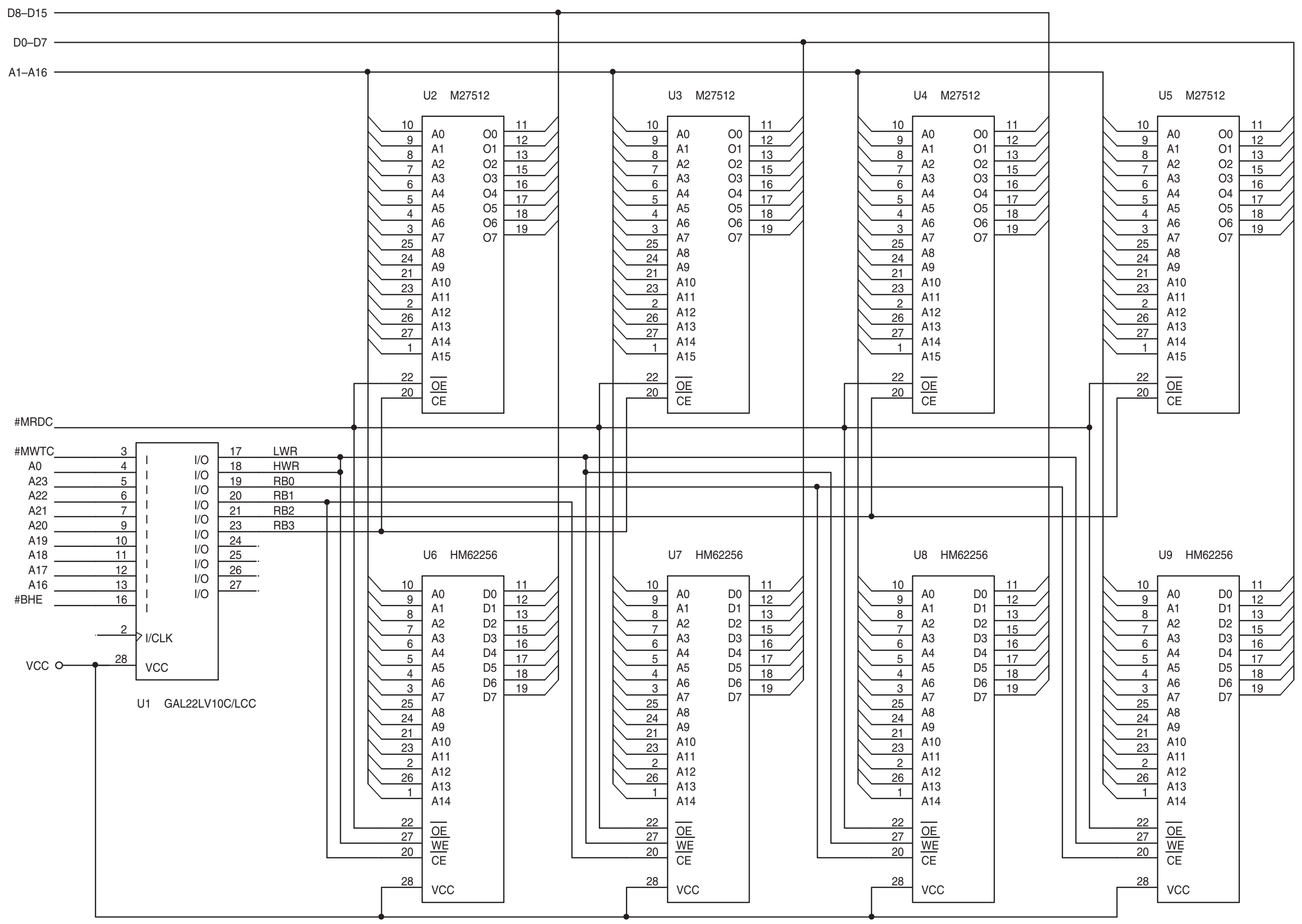


FIGURE 10–32 An 80386SX memory system containing 256K of EPROM and 128K of SRAM.

four 27512 (64K × 8) EPROMs and 128K bytes of SRAM memory found in four 62256 (32K × 8) SRAMs.

Notice in Figure 10–32 that the PLD also generates the memory bank write signals $\overline{\text{LWR}}$ and $\overline{\text{HWR}}$. As can be gleaned from this circuit, the number of components required to interface memory has been reduced to just one, in most cases (the PLD). The program listing for the PLD is located in Example 10–8. The PLD decodes the 16-bit-wide memory addresses at locations 000000H–01FFFFH for the SRAM and locations FC0000H–FFFFFFH for the EPROM.

EXAMPLE 10–8

```
-- VHDL code for the decoder of Figure 10-32

library ieee;
use ieee.std_logic_1164.all;

entity DECODER_10_32 is

port (
       A23, A22, A21, A20, A19, A18, A17, A16, A0, BHE, MWTC: in STD_LOGIC;
       LWR, HWR, RB0, RB1, RB2, RB3: out STD_LOGIC
);

end;

architecture V1 of DECODER_10_32 is

begin
       LWR <= A0 or MWTC;
       HWR <= BHE or MWTC;
       RB0 <= A23 or A22 or A21 or A20 or A19 or A18 or A17 or A16;
       RB1 <= A23 or A22 or A21 or A20 or A19 or A18 or A17 or not(A16));
       RB2 <= not(A23 and A22 and A21 and A20 and A19 and A18 and A17);
       RB3 <= not(A23 and A22 and A21 and A20 and A19 and A18 and not(A17));
end V1;
```

10–5 80386DX AND 80486 (32-BIT) MEMORY INTERFACE

As with 8- and 16-bit memory systems, the microprocessor interfaces to memory through its data bus and control signals that select separate memory banks. The only difference with a 32-bit memory system is that the microprocessor has a 32-bit data bus and four banks of memory, instead of one or two. Another difference is that both the 80386DX and 80486 (both SX and DX) contain a 32-bit address bus that usually requires PLD decoders instead of integrated decoders because of the sizable number of address bits.

Memory Banks

The memory banks for both the 80386DX and 80486 microprocessors are illustrated in Figure 10–33. Notice that these large memory systems contain four 8-bit-wide banks that each contain up to 1G bytes of memory. Bank selection is accomplished by the bank selection signals $\overline{\text{BE3}}$, $\overline{\text{BE2}}$, $\overline{\text{BE1}}$, and $\overline{\text{BE0}}$. If a 32-bit number is transferred, all four banks are selected; if a 16-bit number is transferred, two banks (usually $\overline{\text{BE3}}$ and $\overline{\text{BE2}}$ or $\overline{\text{BE1}}$ and $\overline{\text{BE0}}$) are selected; and if 8 bits are transferred, a single bank is selected.

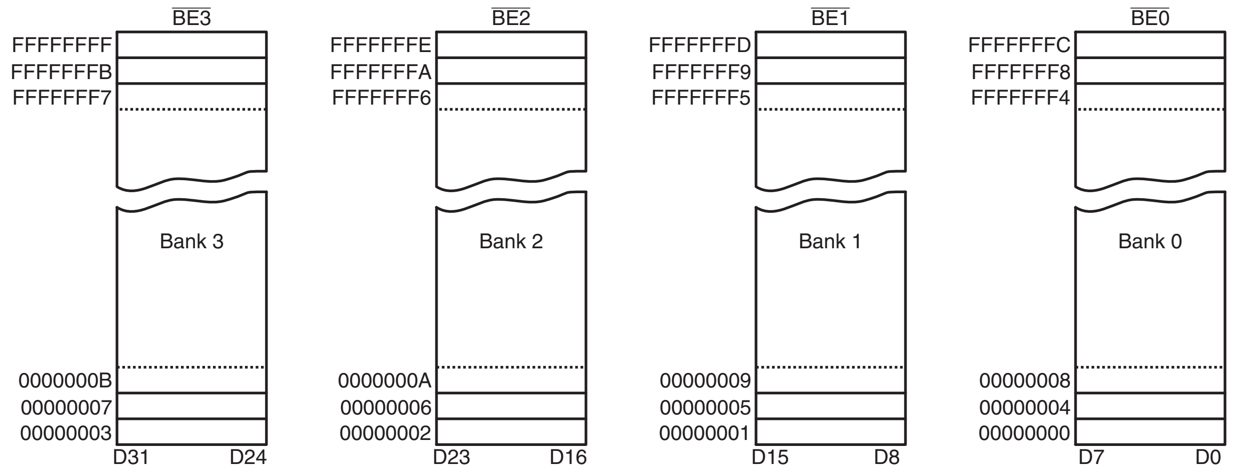
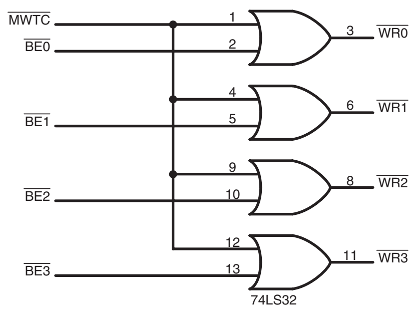


FIGURE 10–33 The memory organization for the 80386DX and 80486 microprocessors.

FIGURE 10–34 Bank write signals for the 80386DX and 80486 microprocessors.



As with the 8086/80286/80386SX, the 80386DX and 80486 require separate write strobe signals for each memory bank. These separate write strobes are developed, as illustrated in Figure 10–34, by using a simple OR gate or other logic component.

32-Bit Memory Interface

As can be gathered from the prior discussion, a memory interface for the 80386DX or 80486 requires that we generate four bank write strobes and decode a 32-bit address. There are no integrated decoders, such as the 74LS138, that can easily accommodate a memory interface for the 80386DX or 80486 microprocessors. Note that address bits A_0 and A_1 are don't cares when 32-bit-wide memory is decoded. These address bits are used within the microprocessor to generate the bank enable signals. Notice that the address bus connection A_2 connects to memory address pin A_0. This occurs because there is no A_0 or A_1 pin on the 80486 microprocessor.

Figure 10–35 shows a 512K × 8 SRAM memory system for the 80486 microprocessor. This interface uses eight 64K × 8 SRAM memory devices, a PLD, and an OR gate. The OR gate is required because of the number of address connections found on the microprocessor. This system places the SRAM memory at locations 02000000H–0203FFFFH. The program for the PLD device is found in Example 10–9.

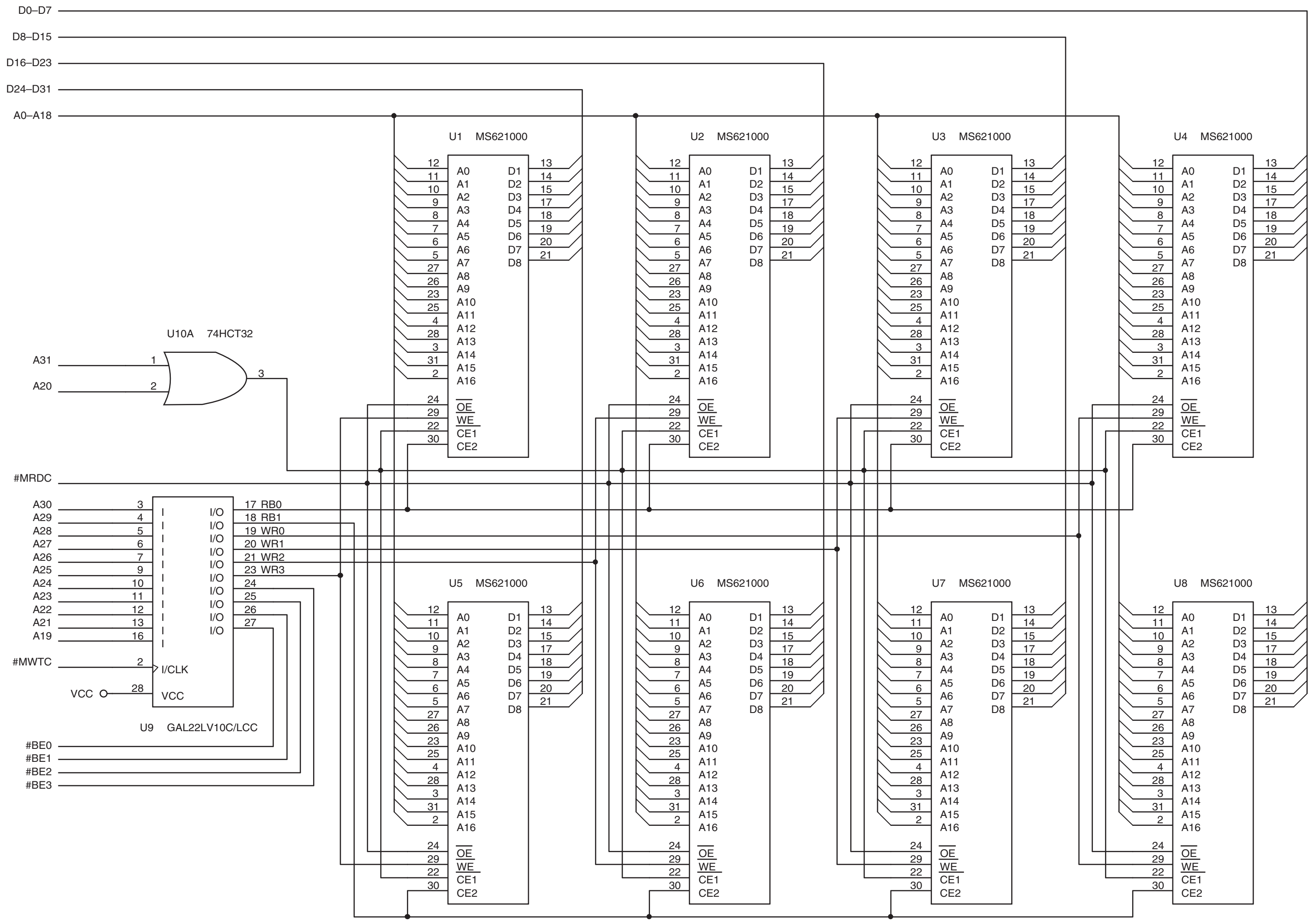


FIGURE 10–35 A small 512K-byte SRAM memory system for the 80486 microprocessor.

EXAMPLE 10–9

```
library ieee;
use ieee.std_logic_1164.all;

entity DECODER_10_35 is

port (
       A30, A29, A28, A27, A26, A25, A24, A23, A22, A21, A19, BE0, BE1, BE2,
             BE3, MWTC: in STD_LOGIC;
       RB0, RB1, WR0, WR1, WR2, WR3: out STD_LOGIC
);

end;

architecture V1 of DECODER_10_35 is

begin

       WR0 <= BE0 or MWTC;
       WR1 <= BE1 or MWTC;
       WR2 <= BE2 or MWTC;
       WR3 <= BE3 or MWTC;
       RB0 <= A30 or A29 or A28 or A27 or A26 or A25 or A24 or A23 or A22
              or A 21 or A19;
       RB1 <= A30 or A29 or A28 or A27 or A26 or A25 or A24 or A23 or A22
              or A 21 or not(A19);

end V1;
```

Although not mentioned in this section of the text, the 80386DX and 80486 microprocessors operate with very high clock rates that usually require wait states for memory access. Access time calculations for these microprocessors are discussed in Chapters 17 and 18. The interface provides a signal used with the wait state generator that is not illustrated in this section of the text. Other devices with these higher speed microprocessors are cache memory and interleaved memory systems. These are also presented in Chapter 17 with the 80386DX and 80486 microprocessors.

10–6 PENTIUM THROUGH CORE2 (64-BIT) MEMORY INTERFACE

The Pentium through Core2 microprocessors (except for the P24T version of the Pentium) contain a 64-bit data bus, which requires either eight decoders (one per bank) or eight separate write signals. In most systems, separate write signals are used with this microprocessor when interfacing memory. Figure 10–36 illustrates the Pentium's memory organization and its eight memory banks. Notice that this is almost identical to the 80486, except that it contains eight banks instead of four.

As with earlier versions of the Intel microprocessor, this organization is required for upward memory compatibility. The separate write strobe signals are obtained by combining the bank enable signals with the $\overline{\text{MWTC}}$ signal, which is generated by combining the M/$\overline{\text{IO}}$ with W/$\overline{\text{R}}$. The circuit employed for bank write signals appears in Figure 10–37. As can be imagined, we often find a PLD used for bank write signal generation.

64-Bit Memory Interface

Figure 10–38 illustrates a small Pentium–Core2 memory system. This system uses a PLD to decode the memory address. This system contains eight 27C4001 EPROM memory devices (512K × 8), interfaced to the Pentium–Core2 at locations FFC00000H through FFFFFFFFH.

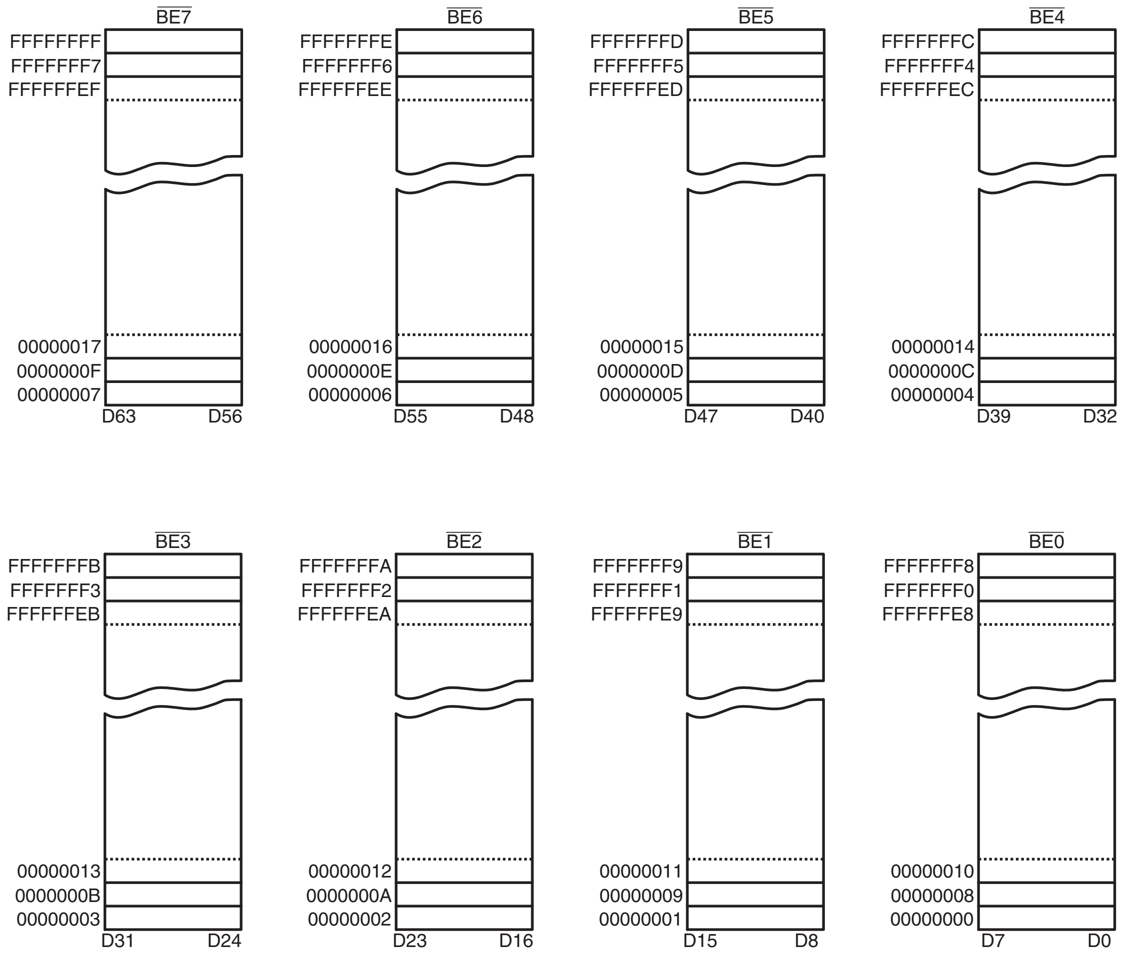


FIGURE 10–36 The memory organization of the Pentium–Core2 microprocessors.

This is a total memory size of 4M bytes organized so that each bank contains two memory components. Note that the Pentium Pro through the Core2 can be configured with 36 address connections, allowing up to 64G of memory. The Pentium 4 and the Core2 can also be configured in the flat mode and may contain up to 40 address connections. (The Core2 contains only 36.)

Memory decoding, as illustrated in Example 10–10, is similar to the earlier examples, except that with the Pentium–Core2 the rightmost three address bits (A_2–A_0) are ignored. In this case, the decoder selects sections of memory that are 64 bits wide and contain 4M bytes of EPROM memory.

The A_0 address input of each memory device connects to the A_3 address output of the Pentium and above. This A_1 address input of each memory device connects to the A_4 address output of the Pentium and above. This skewed address connection continues until the A_{18} address input to the memory is connected to the A_{22} address output of the Pentium. Address positions A_{22}–A_{31} are decoded by PLD. The program for the PLD device is listed in Example 10–10 for memory locations FFC00000H–FFFFFFFFH.

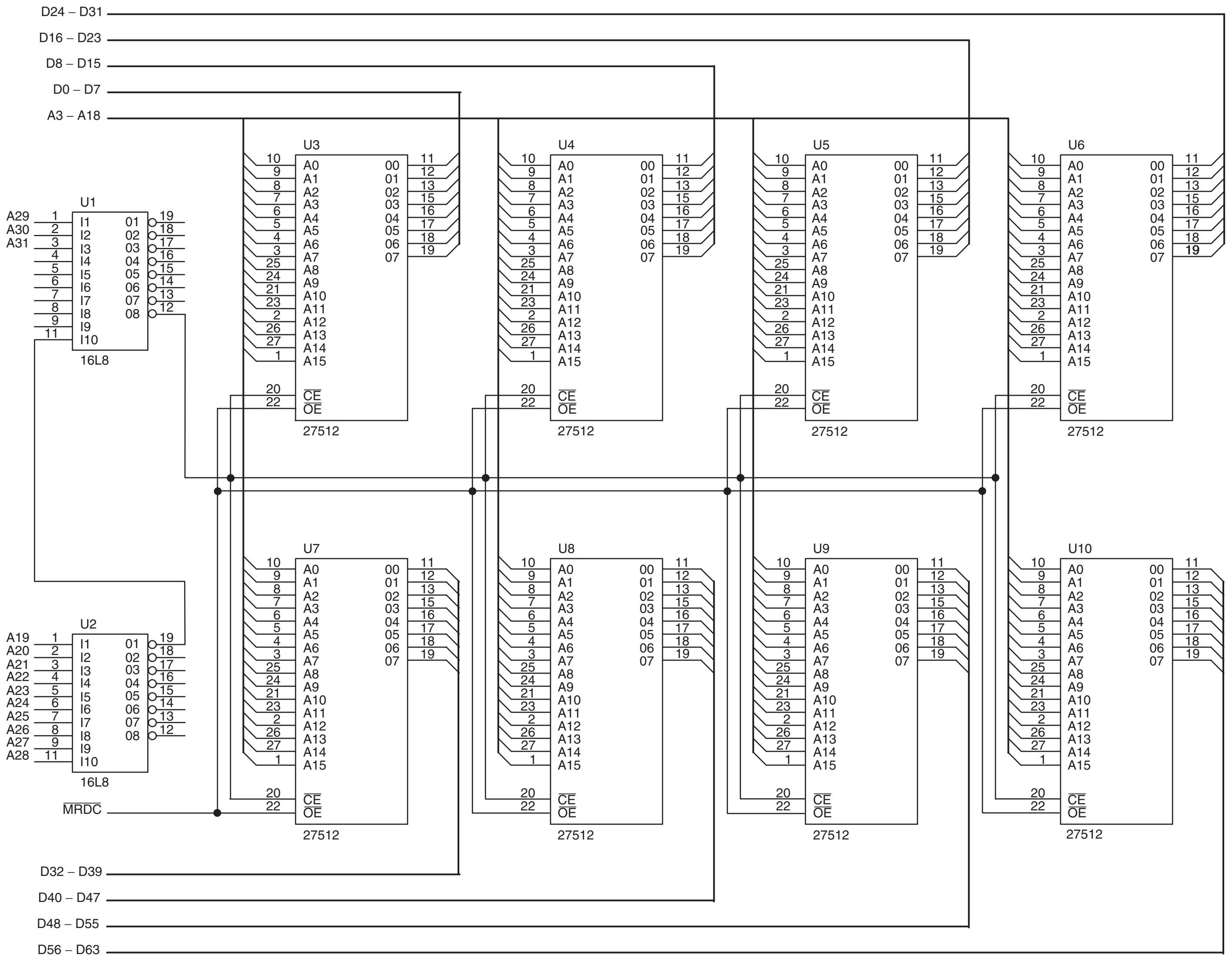


FIGURE 10–37 A small 512K-byte EPROM memory interfaced to the Pentium–Core2 microprocessors.

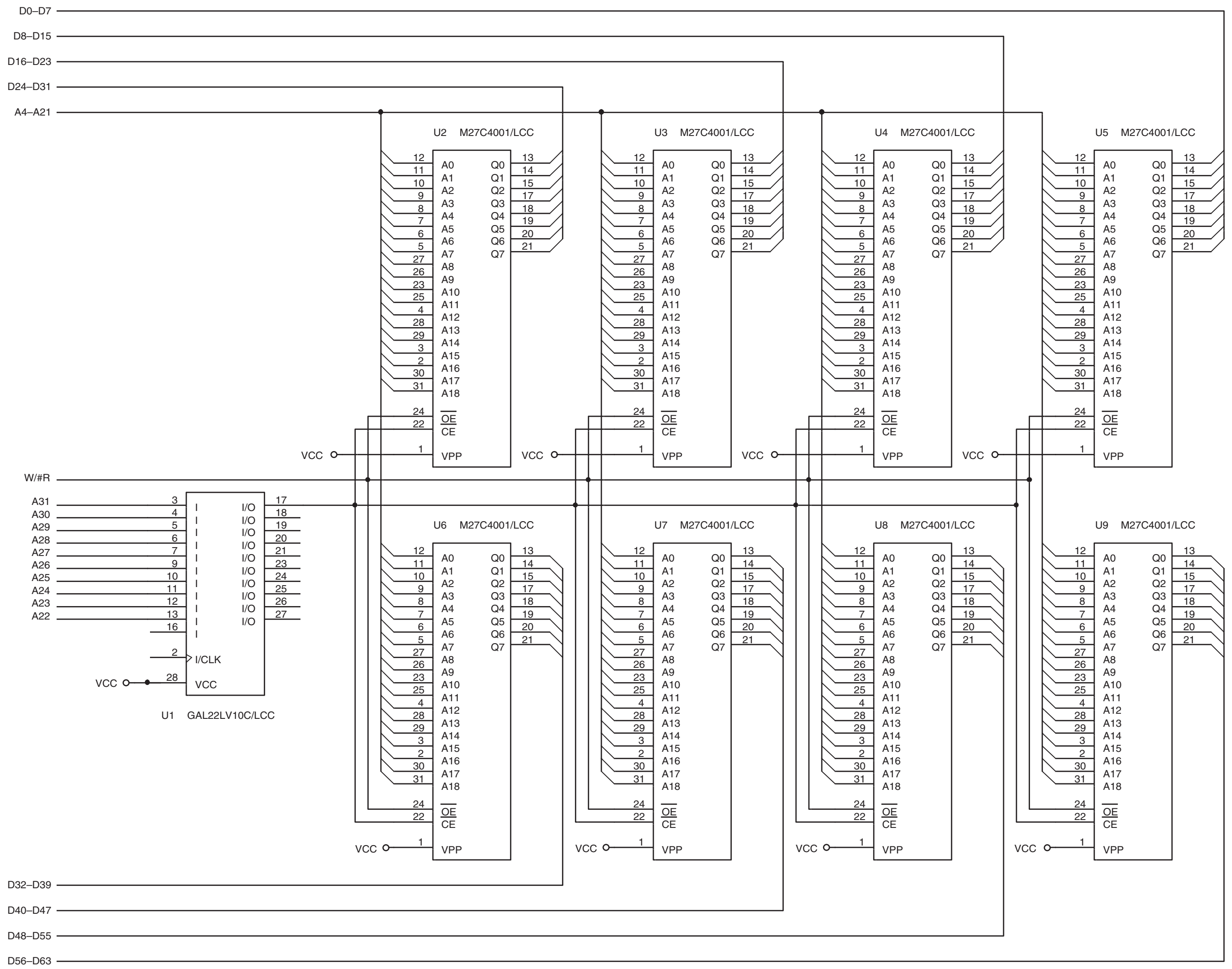


FIGURE 10–38 A small 4M-byte EPROM memory system for the Pentium–Core2 microprocessors.

EXAMPLE 10–10

```
library ieee;
use ieee.std_logic_1164.all;

entity DECODER_10_38 is

port (
       A31, A30, A29, A28, A27, A26, A25, A24, A23, A22: in STD_LOGIC;
       SEL: out STD_LOGIC
);

end;

architecture V1 of DECODER_10_38 is

begin
       SEL <= not(A31 and A30 and A29 and A28 and A27 and A26 and A25 and A24
              and A23 and A22);

end V1;
```

Not explained in this text is the memory interface for an Itanium and Itanium II from Intel, which contains a data bus width of 128 bits. From the information presented in this section of the chapter, it is a fairly easy task to create a memory with 16 banks for the Itanium.

10–7 DYNAMIC RAM

Because RAM memory is often very large, it requires many SRAM devices at a great cost or just a few DRAMs (**dynamic RAMs**) at a much reduced cost. The DRAM memory, as briefly discussed in Section 10–1, is fairly complex because it requires address multiplexing and refreshing. Luckily, the integrated circuit manufacturers have provided a dynamic RAM controller that includes the address multiplexers and all the timing circuitry necessary for refreshing.

This section of the text covers the DRAM memory device in much more detail than in Section 10–1 and provides information on the use of a dynamic controller in a memory system.

DRAM Revisited

As mentioned in Section 10–1, a DRAM retains data for only 2–4 ms and requires the multiplexing of address inputs. Although address multiplexers have already been covered in Section 10–1, the operation of the DRAM during refresh is explained in detail here.

As previously mentioned, a DRAM must be refreshed periodically because it stores data internally on capacitors that lose their charge in a short period of time. To refresh a DRAM, the contents of a section of the memory must periodically be read or written. Any read or write automatically refreshes an entire section of the DRAM. The number of bits that are refreshed depends on the size of the memory component and its internal organization.

Refresh cycles are accomplished by doing a read, a write, or a special refresh cycle that doesn't read or write data. The refresh cycle is internal to the DRAM and is often accomplished while other memory components in the system operate. This type of memory refresh is called either *hidden refresh, transparent refresh,* or sometimes *cycle stealing.*

In order to accomplish a hidden refresh while other memory components are functioning, an $\overline{\text{RAS}}$-only cycle strobes a row address into the DRAM to select a row of bits to be refreshed. The $\overline{\text{RAS}}$ input also causes the selected row to be read out internally and rewritten into the selected bits. This recharges the internal capacitors that store the data. This type of refresh is hidden from the system because it occurs while the microprocessor is reading or writing to other sections of the memory.

The DRAM's internal organization contains a series of rows and columns. A 256K × 1 DRAM has 256 columns, each containing 256 bits, or rows organized into four sections of 64K bits each. Whenever a memory location is addressed, the column address selects a column (or internal memory word) of 1024 bits (one per section of the DRAM). Refer to Figure 10–39 for the internal structure of a 256K × 1 DRAM. Note that larger memory devices are structured similarly to the 256K × 1 device. The difference usually lies in either the size of each section or the number of sections in parallel.

Figure 10–40 illustrates the timing for an $\overline{\text{RAS}}$-only refresh cycle. The difference between the $\overline{\text{RAS}}$ and a read or write is that it applies only a refresh address, which is usually obtained from a 7- or 8-bit binary counter. The size of the counter is determined by the type of DRAM being refreshed. The refresh counter is incremented at the end of each refresh cycle so all the rows are refreshed in 2 or 4 ms, depending on the type of DRAM.

If there are 256 rows to be refreshed within 4 ms, as in a 256K × 1 DRAM, then the refresh cycle must be activated at least once every 15.6 μs in order to meet the refresh specification. For example, it takes the 8086/8088, running at a 5 MHz clock rate, 800 ns to do a read or a write. Because the DRAM must have a refresh cycle every 15.6 μs, for every 19 memory reads or writes, the memory system must run a refresh cycle or else memory data will be lost. This represents a loss of 5% of the computer's time, a small price to pay for the savings represented by using the dynamic RAM. In a modern system such as a 3.0 GHz Pentium 4, 15.6 μs is a great deal of time. Since the 3.0 GHz Pentium 4 executes an instruction in about one-third ns (many instructions execute in a single clock), it can execute about 46,000 instructions between refreshes. This means that in the new machines much less than 1% (≈ 0.002 %) is required for a refresh.

EDO Memory

A slight modification to the structure of the DRAM changes the device into an EDO (**extended data output**) DRAM device. In the EDO memory, any memory access, including a refresh, stores the 256 bits selected by $\overline{\text{RAS}}$ into latches. These latches hold the next 256 bits of information, so in most programs, which are sequentially executed, the data are available without any wait states. This slight modification to the internal structure of the DRAM increases system performance by about 15% to 25%. Although EDO memory is no longer available, this technique is still employed in all modern DRAM.

SDRAM

Synchronous dynamic RAM (**SDRAM**) is used with most newer systems in one form or another because of its speed. Versions are available with access times of 10 ns for use with a 66 MHz system bus; 8 ns for use with a 100 MHz system bus; and 7 ns for the 133 MHz bus. At first, the access time may lead one to think that these devices operate without wait states, but that is not true. After all, DRAM access time is 60 ns and SDRAM access time is 10 ns. The 10 ns access time is misleading because it only applies to the second, third, and fourth 64-bit reads from the device. The first read requires the same number of waits as a standard DRAM.

When a burst transfer occurs to the SDRAM from the microprocessor, it takes three or four bus clocks before the first 64-bit number is read. Each subsequent number is read without wait states and in one bus cycle each. Because SDRAM bursts read four 64-bit numbers, and the second through the fourth require no waits and can be read in one bus cycle each, SDRAM outperforms standard DRAM or even EDO memory. This means that if it takes three bus cycles for the first number and three more for the next three, it takes a total of seven bus clocks to read four 64-bit numbers. If this is compared to DRAM, which takes three clocks per number or 12 clocks,