Semiconductor Memories and Interfacing

8

Objectives

At the conclusion of this chapter you should be able to:

1. *Understand types of memories.*
2. *Know about RAM, ROM, PROM, EPROM, EEPROM, FLASH ROM, NVRAM.*
3. *Understand static and dynamic memories.*
4. *Realize the need for DRAM controller.*
5. *Interfacing ROM, SRAM, DRAM to system bus.*
6. *Understand error detecting and correcting in memories*
7. *Know the current trends, North Bridge, South Bridge.*

Memories are devices into which information can be written to and retrieved whenever required. In this chapter, we shall study various categories of semiconductor memory devices used in a computer system and their interfacing to a system bus. There are secondary memory devices such as magnetic disks and optical disks not discussed here. First-generation computers used relays as memories and second-generation computers used magnetic core memories. All these are obsolete now. From the third generation to present times, computers have used semiconductor memories. There are various categories of memories which are explained here.

TYPES OF MEMORIES

RAM—Random Access Memory

The name is given by the manufacturer but actually it should be Read/Write Memory (RWM). It is volatile meaning that information in the memory is lost/unpredictable when power is removed. The types of RAM that are available are bipolar static RAM, static MOS RAM, Dynamic MOS RAM, Pseudo Static MOS RAM, EDO (Extended Data Output) RAM, Synchronous DRAMs, Rhombus DRAMS, and DDRAMs, etc. In future, FRAMs (Ferro Electrical RAMs) may be available.

ROM—Read Only Memory

This is also called One Time Programmable (OTP). Once programmed, it is permanent and not reversible. ROMs are developed by the manufacturers and not programmed by the user. For heavy volume production, ROMs turn out to be very cheap compared to other non-volatile memories like PROMs, EPROMs, EEROMs, and Flash ROMs.

PROM

It is similar to ROM with a difference that this can be one time programmable by the user using a programming device and not by the manufacturer; hence, it will be very useful for prototyping purposes. Examples are 27PC32 (4k × 8), 27PC64 (8k × 8), 27PC128 (16k × 8), 27PC256 (32k × 8), 27PC512 (64k × 8), 27PC010 (128k × 8), etc.

EPROM

It belongs to the ROM family and has a window on the chip. Once it is programmed, the user has to take precautions not to expose it even to sunlight. Then contents will stay for approximately 10 years and in this sense, it can be classified as non-volatile memory. Whenever you want to rewrite, the EPROM has to be taken out of the circuit board, the window on the chip is to be exposed to ultraviolet light for approximately 5 to 10 minutes. This process erases the entire information on the chip and the user have to program the entire information on a separate programmer. After this is done, the window on this chip is to be closed with a tape and the EPROM can be plugged on the circuit board. Examples of EPROM are 2708 (1k × 8), 2716 (2k × 8), 2732 (4k × 8), 2764 (8k × 8), 27128 (16k × 8), etc.

EEPROM

It is different from the EPROM because EEPROM can be programmed in-circuit. This will not have a window. In EEPROM, you can erase a bit and write that bit and this can be done on the board itself. Each location can be written only 10,000 times. In this sense, it is not perfectly non-volatile like EPROM. Programming of the EEPROM takes more time than reading; so it cannot substitute RWM (RAM). Examples are 2816A (2k × 8), 2817A (2k × 8), 28C84 (8k × 8).

Flash ROM

Flash ROM is similar to EEPROM but has fixed block sizes, typically of 128–512 k bits, which can be erased at a time. This is the most widely used non-volatile memory nowadays. Instead of each bit, you can erase a block and re-write the erased block. Flash memory is widely used in memory cards for digital cameras, mobile phones, computer memory sticks, and the like. Examples are 28F256 (32k × 8), 28F512 (64k × 8), 28F400 (256k × 16 or 512k × 8), 28F008 (1 M × 8), 28F016 (1M × 16) or (2M × 8), 28F032 (2M × 16 or 4M × 8). Now higher capacity USB Flash memories with are available. They are called pen drives (memory stick) with capacities of 2 GB, 4 GB, 8 GB, etc.

Shadow RAM or NVRAM

When power is on, it works like the normal ROM but the moment it falls below +5 V, it transfers everything to the shadow EEPROM. When the power comes back, the reverse process is done and the contents are transferred back to RAM. In this way, it works as a non-volatile RAM. The cost of this RAM is very high. It has a shadow EEPROM on the chip itself. Because of its cost, it is now obsolete.

RAM

This is a read and write memory and is also volatile. There are two types of RAMs, *static* RAM and *dynamic* RAM. Static RAM uses a flip-flop as a basic cell to store one bit of information. It is available in bipolar and MOS technology. Dynamic memory uses a simple capacitor and the charge or no-charge indicates the two states of a bit. This is available only in MOS technology. Because the basic cell is a capacitor, the charge on the capacitor will either discharge or a discharged capacitor may acquire some charge. Hence, dynamic memories require the REFRESH technique which Static RAMs do not need. Categories of available RAMS are discussed.

STATIC RAM (SRAM)

Static RAM is costly compared to other DRAMS. The SRAM takes about four times the space for a given number of bits than the dynamic RAM, but will not need to refresh the memory cell as is required in DRAM and is, therefore, faster to access. It is mentioned that the typical access time of SRAM would be 25 nanoseconds in contrast to a typical access time of 60 nanoseconds for the dynamic RAM. Of course, DRAM access times have improved due to the recent development in technology. For level-1 and level-2 caches, SRAMs are preferred over superscalar microprocessors. Currently, a 265K × 16, 3.3 V SRAM with access time of 15 ns is available.

BURST (OR SYNCHBURST) STATIC RAM (BSRAM)

SRAMs synchronized either with the system clock or the cache bus clock are known as Burst SRAMs (also refered to as SynchBurst SRAMs). This permits it to be more easily synchronized with any unit that accesses it and reduces the access waiting time. The I Pentium II microprocessor used this as the external level-2 cache memory.

DYNAMIC RAM (DRAM)

Dynamic RAM basic cell is a capacitor that needs frequent charge refreshing as described above to retain its charge. The DRAM must be refreshed as per manufacturer-specified time intervals and is less expensive than SRAM. The DRAM bandwidth requirements typically double every three years.

FAST PAGE MODE DRAM (FPM DRAM)

In DRAM, the RAS/ is first sent out with DRAM row address and this process selects the row in DRAM. Then CAS/ is sent with column address to select the column and thus the cell. The state of the cell is sent out in the READ operation. If the RAS/ signal is kept active and CAS/ is sent several times with different column addresses, all the row cells can be read. This mode is called Fast Page Mode or Fast Page Mode DRAM. This results not only in reduction of access time but also lowers power requirements.

ENHANCED DRAM

A combination of SRAM (typically 256 bytes) and DRAM in a single package is known as Enhanced DRAM (EDRAM). EDRAM is usually used for a level-2 cache. Data is read first from the faster SRAM and if it is not there, it is read from DRAM.

EXTENDED DATA OUTPUT RAM OR DRAM (EDO RAM OR EDO DRAM)

EDO DRAM is similar to Fast Page Mode DRAM with the additional feature of starting a new access cycle while maintaining the data output of the previous cycle. This creates a

certain amount of overlap in operation (pipelining) resulting in somewhat improved speed. This would be 5% faster than Fast Page Mode DRAM and up to 25% faster than the standard DRAM.

BURST EXTENDED DATA OUTPUT DRAM (BEDO DRAM)

The BEDO DRAM (Burst Extended Data Output DRAM) is a member of the DRAM family which can send out data from one READ operation while at the same time it would be reading the address of the next data to be sent. Also, after reading the address, it can send the data in three successive clock cycles without clock coordination. In other words, three successive outputs seem to be sent from the RAM in a sudden burst.

NONVOLATILE RAM (NVRAM)

Already explained under Shadow RAM

SYNCHRONOUS DRAM (SDRAM)

The synchronous DRAM (SDRAM) is synchronized with the clock speed that the system bus is optimized for. The speed of the SDRAM is specified in MHz rather than in nanoseconds (ns). This makes it easier to compare bus speed and the RAM chip speed. You can convert RAM clock speed to nanoseconds by dividing the chip speed into 1 billion ns (one second). For example, an 83 MHz RAM is equivalent to 12 ns. The SDRAM capacity is 32M × 16, operating at 3.3 V and acces time of 7.5 ns is available.

DUAL DATA RATE DRAMs (DDR) FAMILY

Double-Data-Rate Synchronous Dynamic Random Access Memories (DDRs) are very high-speed memories since they use both rising and falling edges of the clock signal to transfer data. Their architecture is same as that of the SDRAM. Thus, a computer system with a clock of 133 MHz running *DDR SDRAM* will essentially perform like a 266 MHz machine. DDR2 is the second generation of DDR memory family and begins with a speed level of 400 MHz as the lowest available while the 400 MHz speed is the highest speed for DDR1. DDR3 is the third generation in the DDR family. It starts at a speed of 800 Mbps and goes up to 1600 Mbps with a bus speed of 2000 MHz. The DDR3 operates at 1.5 V and thus dissipates less power than DDR2 which operates at 1.8 V.

DIRECT RAMBUS DRAM (DRDRAM)

The Direct Rambus DRAM (DRDRAM) is a proprietary technology proposed by *Rambus*, Inc. in partnership with Intel. Like the SLDRAM, it promises RAM speed up to 800 MHz. It has a smaller bus width (16 bits compared to 64 bits) than the current SDRAM designs.

DRAM technologies are summarized in Table 8.1.

Table 8.1 DRAM Technologies

DYNAMIC RAM (DRAM) MEMORY TECHNOLOGIES					
Type	*Year of Intro.*	*Maximum Clock Rate*	*Bus Width*	*Peak Bandwidth*	*Volts*
FPM	1990	25MHz	64 bits	200 MBps	5v
EDO	1994	40MHz	64 bits	320 MBps	5v
SDRAM	1996	133MHz	64 bits	1.1 GBps	3.3v
RDRAM	1998	400MHz (×2)	16 bits	800 MBps	2.5v
DDR SDRAM	2000	266MHz (×2)	64 bits	4.2 GBps	2.5v
DDR2 SDRAM	2003	533MHz (×2)	64 bits	8.5 GBps	1.8v
DDR3 SDRAM	2007	800MHz (×2)	64 bits	12.8 GBps	1.5v

From Computer Desktop Encyclopedia

MEMORY DEVICE CONFIGURATIONS

All READ ONLY MEMORIES (ROMs) have address lines, Chip Select, Chip Enable, and Data Out signals. The number of address lines depend on the number of locations the ROM has. Unless CS/CE is (are) activated, the memory does not do any function. Output of the chip will be tri-stated if CS/CE is not enabled. There could be one or more Chip selects (CS) and Chip Enable that help in the elimination of the address decoding circuitry for minimum system configuration. Sometimes instead of CS, the memory chip will be provided with Chip Enable inputs (CE) having similar functions. These chip select/chip enable signals can be active high or active low. The READ operation of the chip can be done only if the chip is selected by activating the chip select/enable signals. Some chips may also have the output enable signal OE. PROMs, EPROMs, EEROMs, and FLASHROMs will all have signals that are mentioned for ROM and also some additional signals to program PROM<EPROM and EEROM. In the programmed mode for writing information, DATA lines will become inputs. Typical ROM, PROM, EPROM, and EEROM chips are shown in Fig. 8.1.

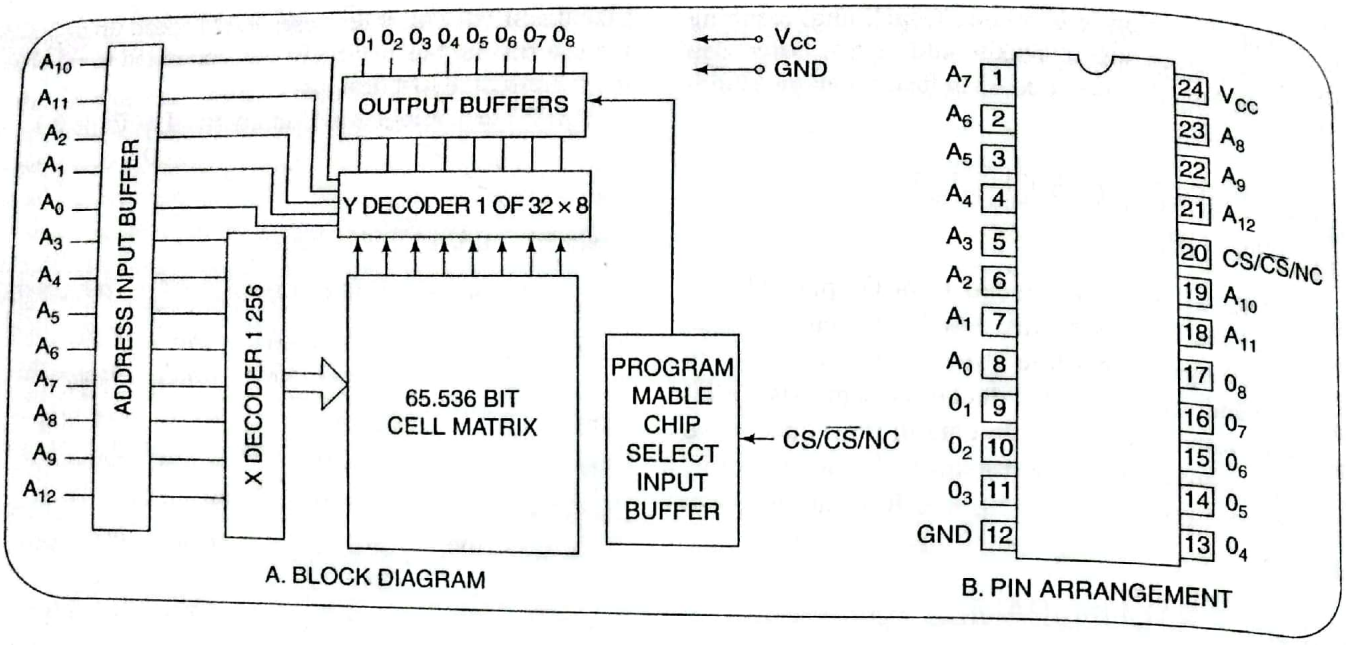


Fig. 8.1 (a) Block diagram of a ROM.

35NVM028

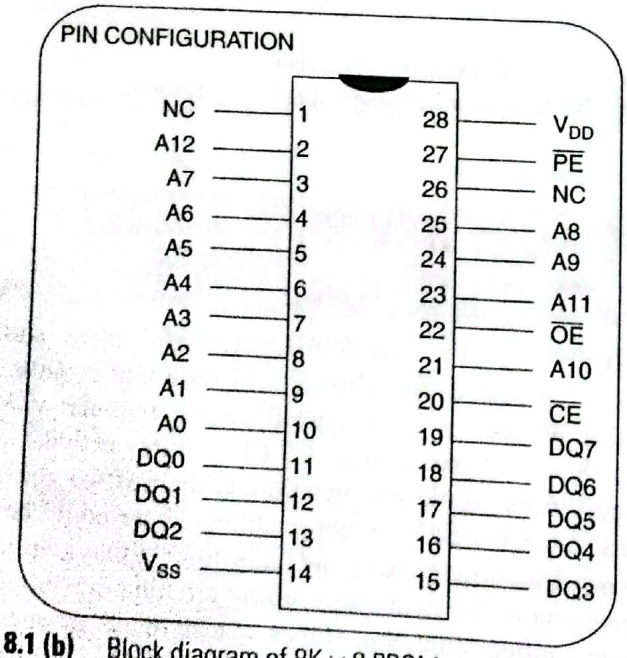


Fig. 8.1 (b) Block diagram of 8K × 8 PROM.

A Static RAM will have all the signals that ROMs have and it will also have the READ/WRITE signal. The data inputs will be either bidirectional and they may have separate DATA IN and DATA OUT lines. A typical 4K × 1 staitc RAM is shown in Fig. 8.2 (a) and 4K × 16 SRAM is shown in Fig. 8.2 (b).

When the memory is not selected, it is not active, draws less current and dissipates very little power. This mode is known as the stand-by mode.

Table 1. Device Operation Truth Table[1]

OE	*PE*	*CE*	*I/O MODE*	*MODE*
X	1	1	Three-state	Standby
0	1	0	Data Out	Read
1	0	0	Data In	Program
1	1	0	Three-state	Read[2]

Notes:
1. "X" is defined as a "don't care" condition.
2. Device active; outputs disabled.

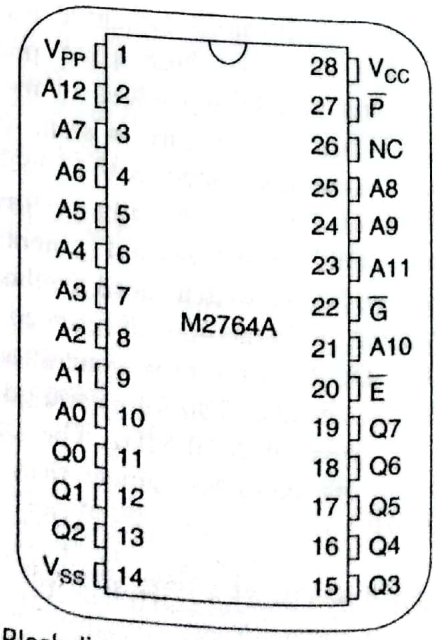
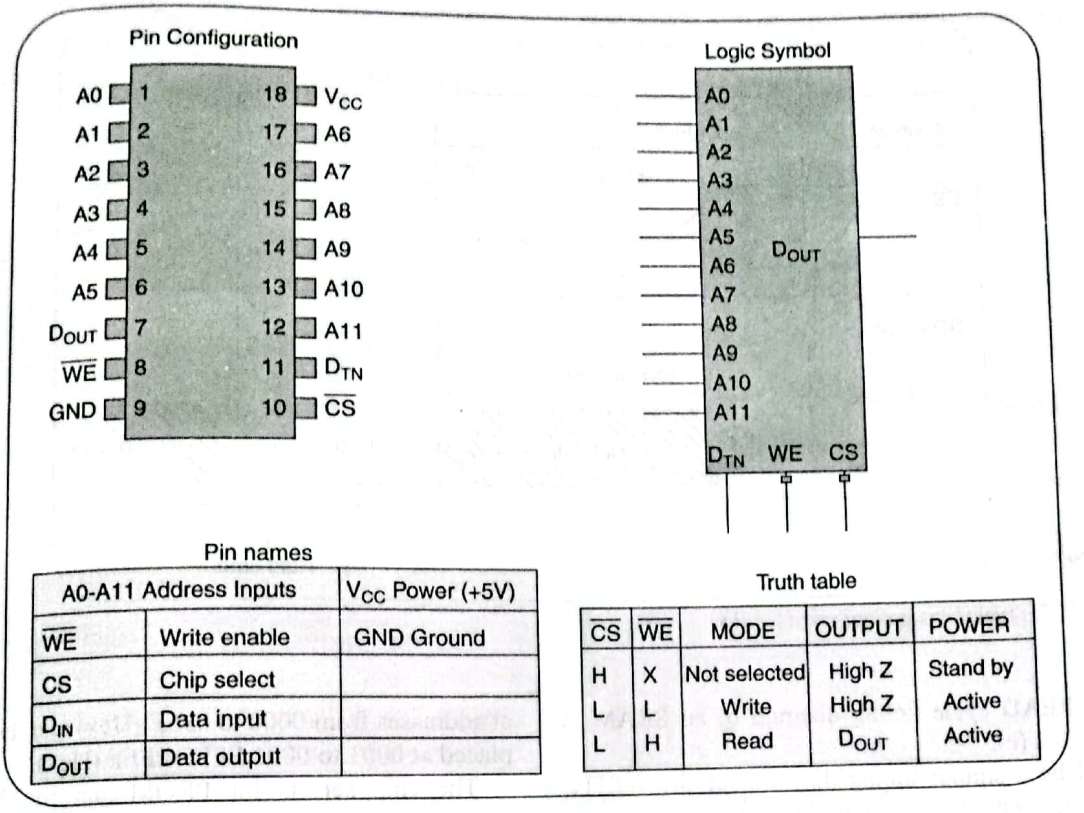


Fig. 8.1 (c) Block diagram of a 8K × 8 EPROM.



Pin names

A0-A11 Address Inputs		V_{CC} Power (+5V)
$\overline{WE}$	Write enable	GND Ground
CS	Chip select	
D_{IN}	Data input	
D_{OUT}	Data output	

Truth table

$\overline{CS}$	$\overline{WE}$	MODE	OUTPUT	POWER
H	X	Not selected	High Z	Stand by
L	L	Write	High Z	Active
L	H	Read	D_{OUT}	Active

Fig. 8.2 (a) 4K × 1 SRAM.

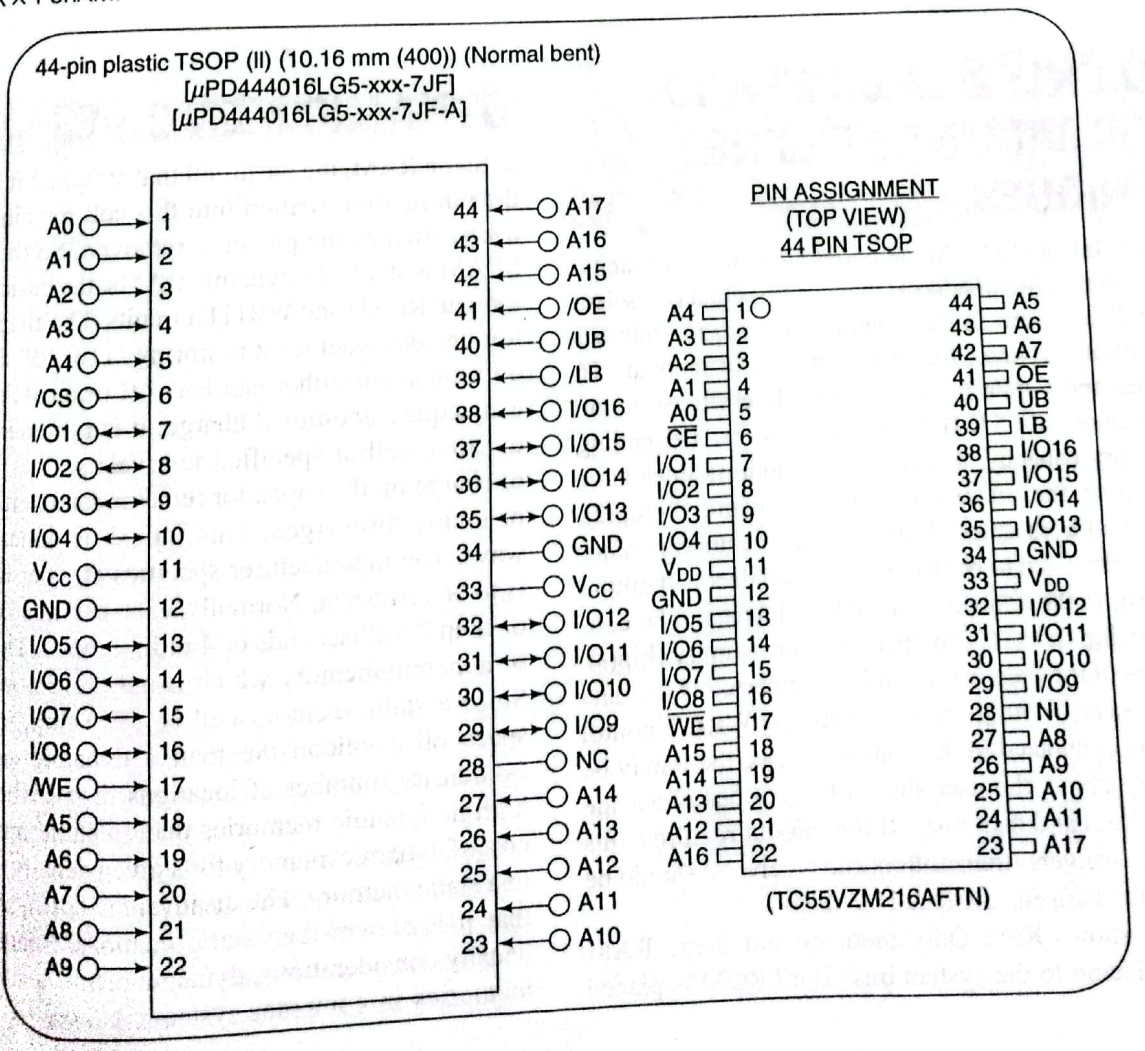


Fig. 8.2 (b) 4K × 16 SRAM.

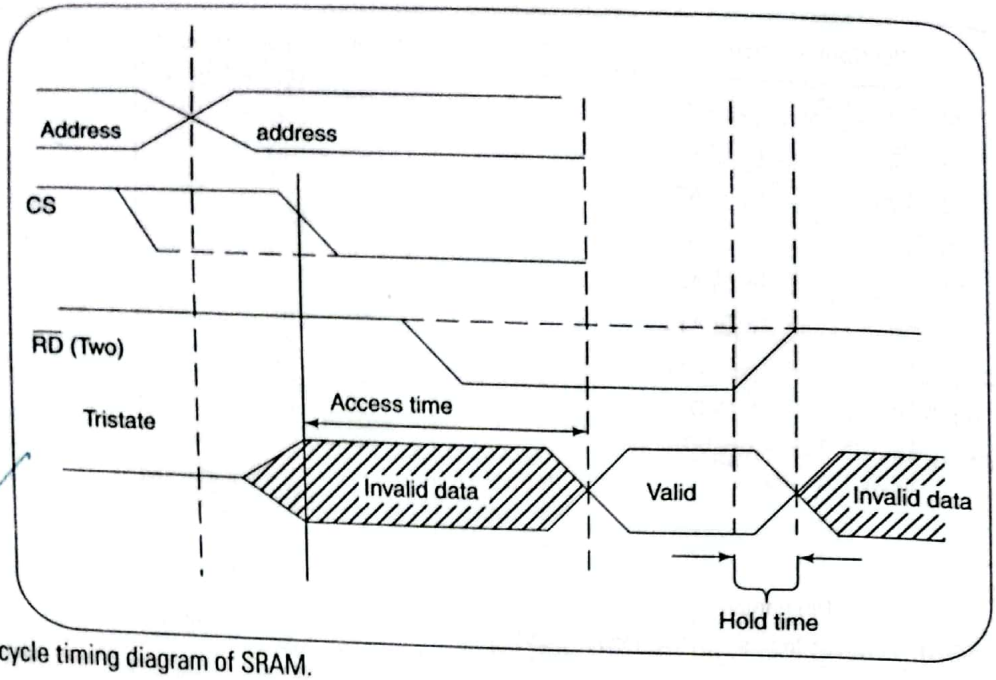


Fig. 8.2 (c) READ cycle timing diagram of SRAM.

A typical READ cycle timing diagram of an SRAM is shown in Fig. 8.2 (c).

There will be a similar timing diagram for the WRITE cycle. This is given as an exercise to the reader.

READ ONLY MEMORIES AND STATIC RAM INTERFACING TECHNIQUES

When Read Only and SRAM memories are of $n \times m$ capacity, where n is the number of addressable bus lines that can select any of the 2^n locations in the memory and m is the number of data input/output lines, for an 8-bit processor there are 16 address lines and 6 data lines. With 16 address lines, it can address a memory of 65 K Byte (8 bits) locations. Depending on the memory capacity, that number of address lines have to connected to the address pins of the memory chip(s). Based on the address space allotted to this memory, remaining address lines need to be decoded and the decoded output corresponding to the address space allotted needs to be connected to the Chip Select (Chip Enable) of the memory chip(s). The data lines of the memory should be connected to the data bus. If it is a Read/Write memory, the READ/WRITE contol line should be connected to the memory. Since there may be many memory chips, the user should be cautious about the fan-out of address and data lines. If the fan-out requirements are not met, transeivers (transmitters and receivers) should be connected to the address and data bus lines.

Figure 8.3 shows Read Only memory and Static RAM memory interfacing to the system bus. The PROM is placed at addresses from 0000 to 03FF (Hex) and the Static RAM is placed at 0001 to 0001 FFFF FFFF (Hex).

The chip selects for PROM and SRAM are decoded accordingly as shown in Fig. 8.3.

DYNAMIC MEMORIES

In Static RAM, the basic cell that stores a bit is a flip-flop. So the information written into this cell remains till it is either overwritten or the power is removed. So the information in SRAM is stable. In dynamic RAMs, the basic cell is a capacitor with READ and WRITE circuits. The charge on the capacitor decides whether it is storing '1' or '0'. Since the charge on a capacitor either can leak off or a discharged capacitor can acquire additional charge, it is necessary to refresh the dynamic cell at specified intervals so that the information or charge on the capacitor remains same, either it is charged or totally discharged. This refresh is done in a number of ways. The manufacturer specifies at what intervals the cell is to be refreshed. Normally, each cell needs to be refreshed once in 2 milliseconds or 4 milliseconds. This is one feature of dynamic memory which is not present in static memory. Since a static memory cell is a flip-flop, it occupies more space on a silicon die than a dynamic cell, a capacitor. So density (number of locations in the silicon die) is very high in dynamic memories than in static memory. Also, the cost of dynamic memory for a given density is low compared to a static memory. The disadvantage of dynamic memory is that it is slower than static memory. Because of cost and density considerations, dynamic memories are used as main memories in computer systems. Now new memories have

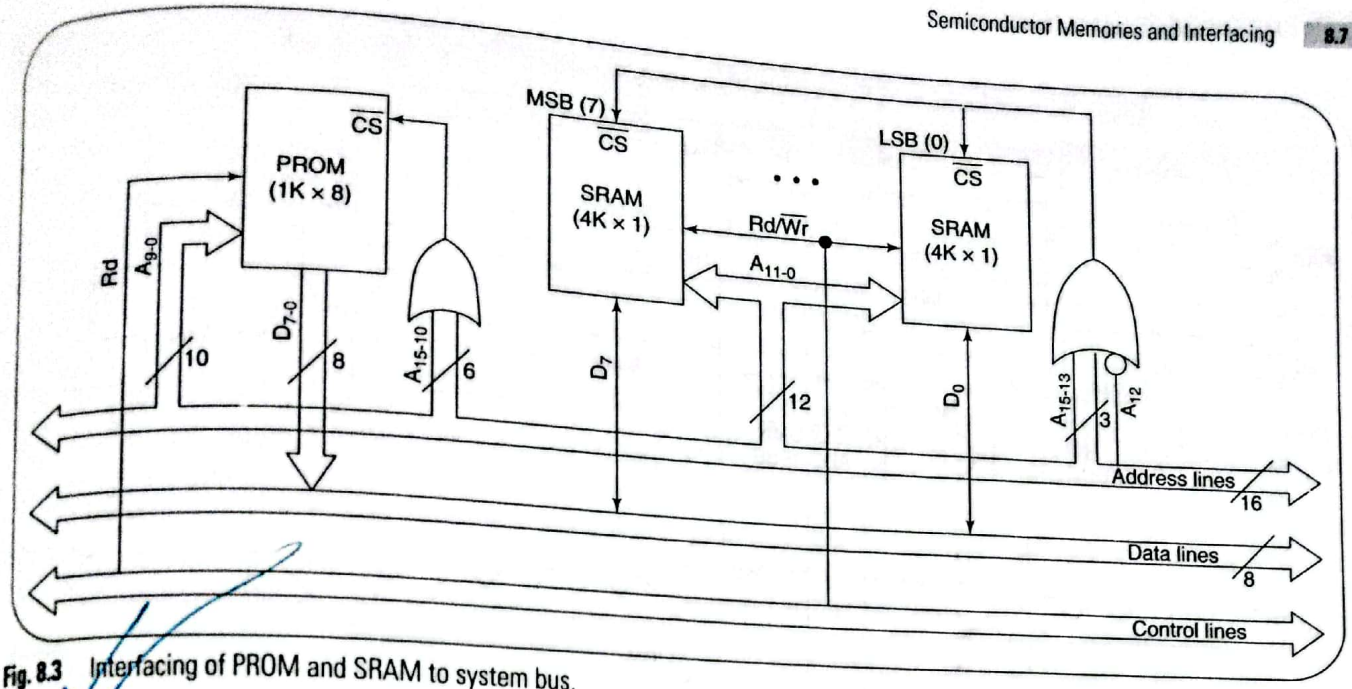


Fig. 8.3 Interfacing of PROM and SRAM to system bus.

been developed and they are known as synchronous DRAMs and Dual Data Rate DRAMS (DDRs). Since the densities of dynamic memories are very high, all address lines to select any location in the DRAM cannot be given at one time. So the addresses are given in a multiplexed way. For these above-mentioned reasons, interfacing a DRAM to a system bus is different from interfacing ROMs and SRAMs. For DRAM interfacing, another controller, a DRAM controller, is needed to provide multiplexed address and do periodic autorefresh.

Bus lines are connected to the DRAM controller and DRAM controller provides the necessary signals to the DRAM. All address lines and chip selects are given to the DRAM controller. The controller, once selected by the chip select, provides the first part of the address, the ROW address, and gives the signal ROW ADDRESS SELECT (RAS/). This address is normally half the total address given to the DRAM controller. This ROW address has to be latched inside the DRAM chip with the RAS/signal. Then the controller gives the remaining part of the address called the column address and gives the CAS/control signal. With RAS and CAS, the required cell is selected and data is either sent out or written in to, depending on the READ/WRITE signal. In case of write, data will be provided to DRAM. Either READ or WRITE operation of the DRAM refreshes that cell.

Refreshing of cells are done row-wise. The refresh time specified by the manufacturer is divided by the number of rows and this time interval is the row refresh time. A timer in the DRAM controller indicates that the row refresh time interval is over. The DRAM controller, if not in the process of reading or writing at that time, sends the row address from an internal row address counter and issues only RAS/ to refresh that row and increment the row address counter. On the expiry of the next row refresh time interval, the incremented row address is refreshed. In this way, it refreshes the entire DRAM cells in the manufacturer specified refresh time. Between refresh time intervals, the DRAM is made available to the system.

Figure 8.4 (a) shows a 64K × 4 DRAM chip with pin details with all signals.

Typical WRITE and READ cycles in DRAM are shown in Fig. 8.4 (b) and 8.4 (c).

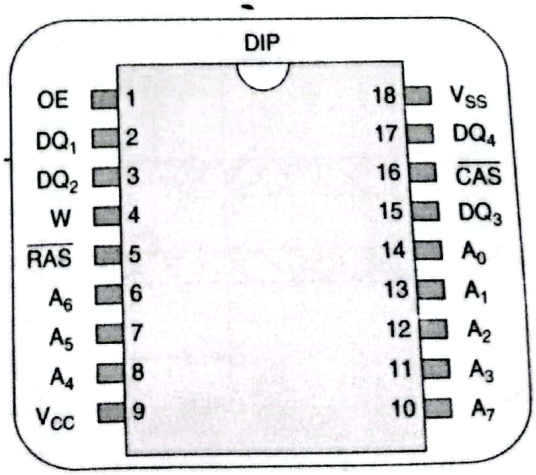


Fig. 8.4 (a) 64K × 4 DRAM.

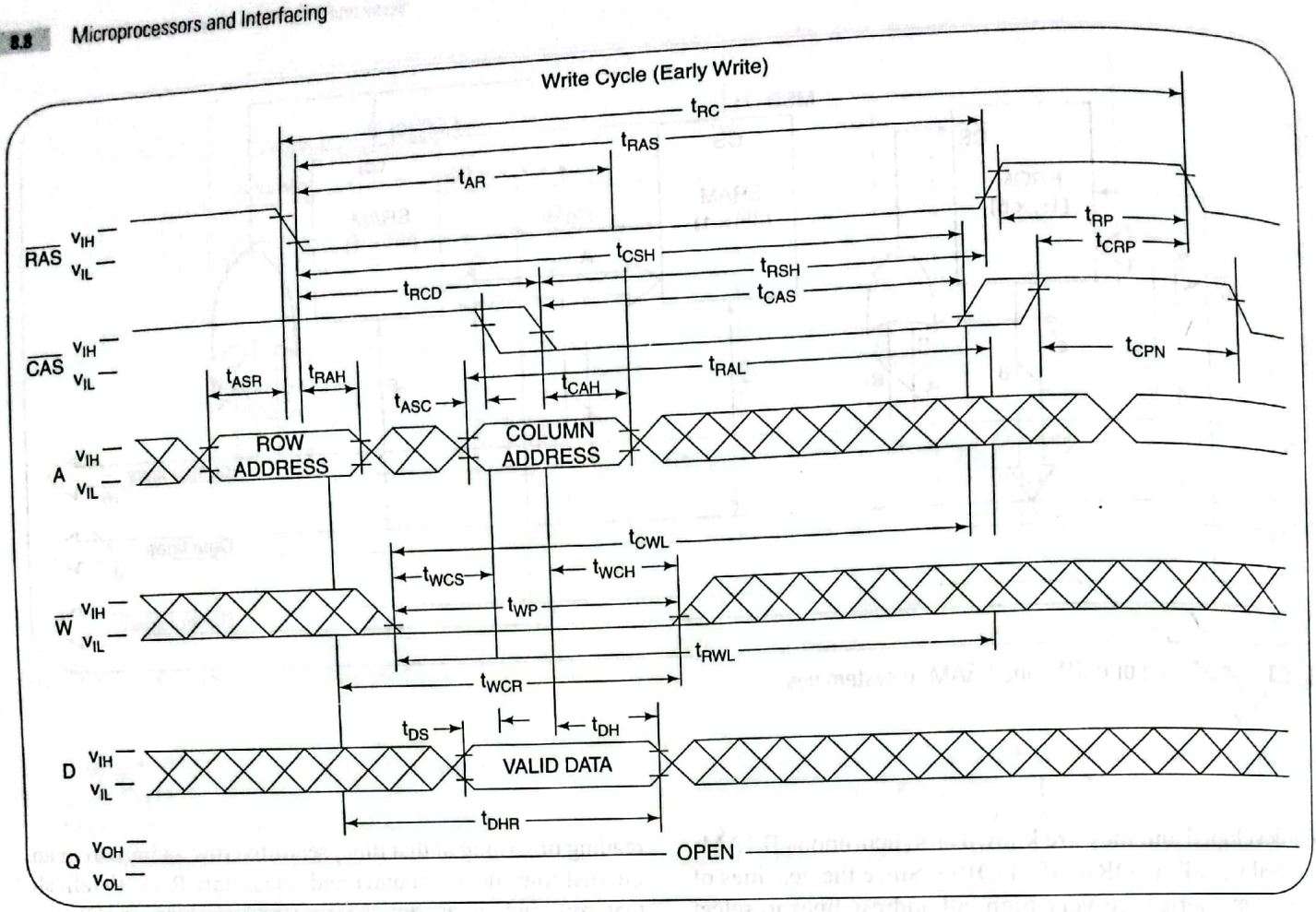


Fig. 8.4 (b) Memory WRITE machine cycle.

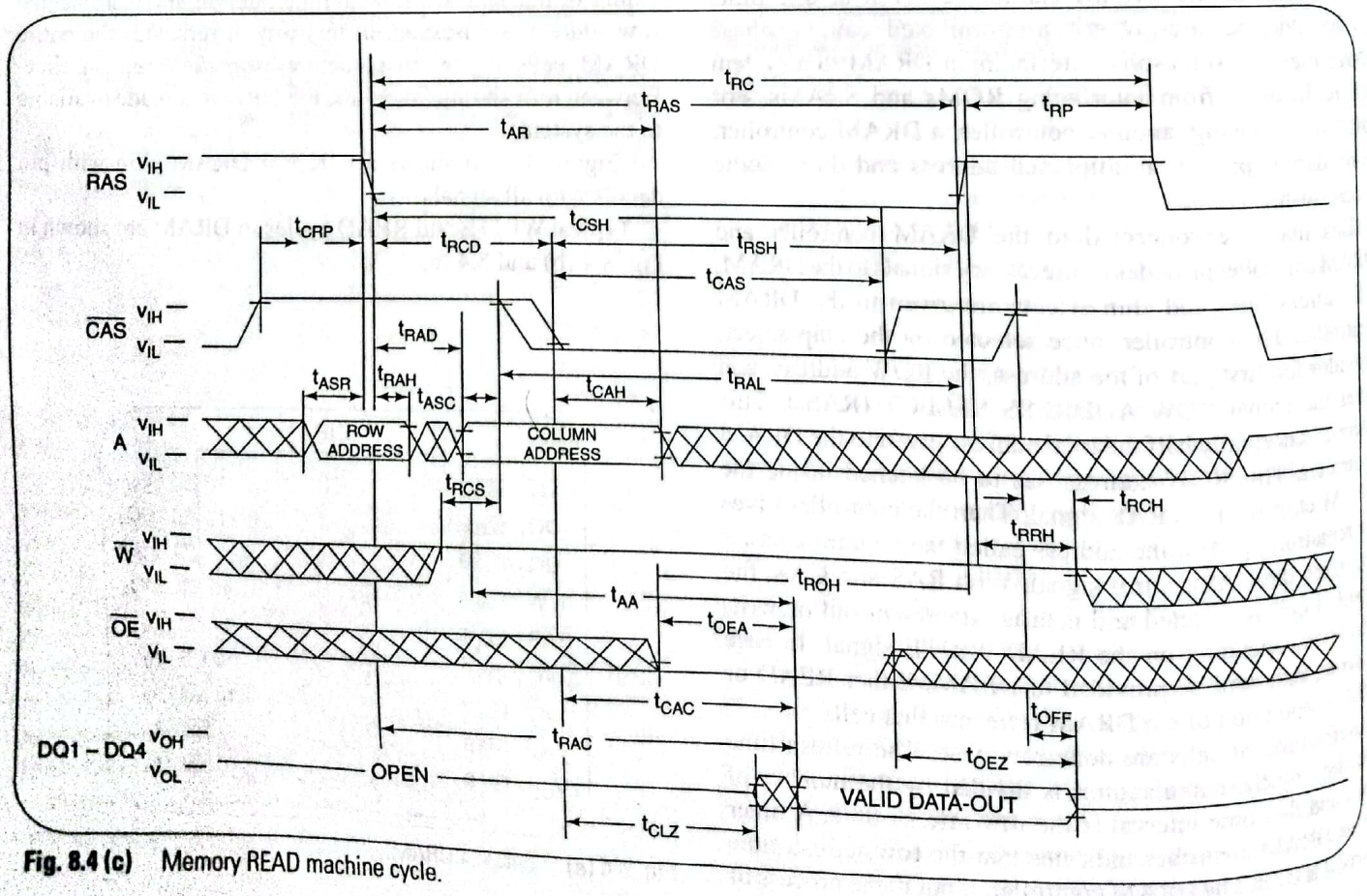


Fig. 8.4 (c) Memory READ machine cycle.

DRAM INTERFACING: TO THE SYSTEM BUS

Now interfacing a 16K × 1 DRAM chip [Fig. 8.5 (a)] to a system bus will be discussed. The DRAM controller block diagram is depicted in Fig. 8.5 (b).

16 address lines from the system bus go to the DRAM controller. This DRAM is placed at addresses 4000 to 7FFF. The most significant two bits of the address bus (A15 and A14) are decoded and given to the chip select of the DRAM controller. A13 and A12 are given to the block selection inputs of DRAM controller.

Now we discuss interfacing DRAM memories to the 8086 processor and also the need for error checking and possibly correcting the error of the information retrieved from memories.

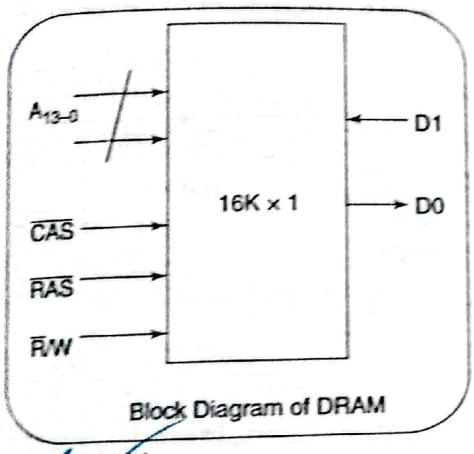


Fig. 8.5 16K × 1 DRAM.

USING AN 82C08 DRAM CONTROLLER IC WITH AN 8086

In high-performance systems where we need DRAM refreshing to take up a minimum amount of the processor's time, we usually use a dedicated device that handles all of refreshing chores without tying up the microprocessor or its buses as the DMA approach. Another example of this type of device is Intel 82C08. Figure 8.8 shows, in block diagram form, how 82C08 can be connected with an 8086 in maximum mode to refresh and control 512 Kbytes of dynamic RAM. 82C08 takes care of all of the addressing and refresh tasks we described before.

The memories here are 256K × 4 devices. As usual for an 8086 system, the memory is set up as 2-byte-wide banks. In this system, each bank has two DRAM devices, and so each bank has 256 Kbytes.

One important point to observe here is that the status signals, S0–S3, from the 8086 are connected directly to the control inputs of the 82C08. Which decodes these status signals to produce the read and write signals needed for the DRAMs. This advanced decoding means that, except when a refresh cycle is in progress, the 8086 can read a byte or word from the DRAMs without WAIT states.

If you look closely at 82C08 in Fig. 8.8, you should find the port enable input, $\overline{PE}$. This input is asserted low to request access to DRAM. If 82C08 is not involved in a refresh operation when $\overline{PE}$ is asserted low, 82C08 will multiplex the address from the address bus into the DRAMs with appropriate $\overline{RAS}$ and $\overline{CAS}$ strobes. 82C08 will also send out an $\overline{AACK}$ signal

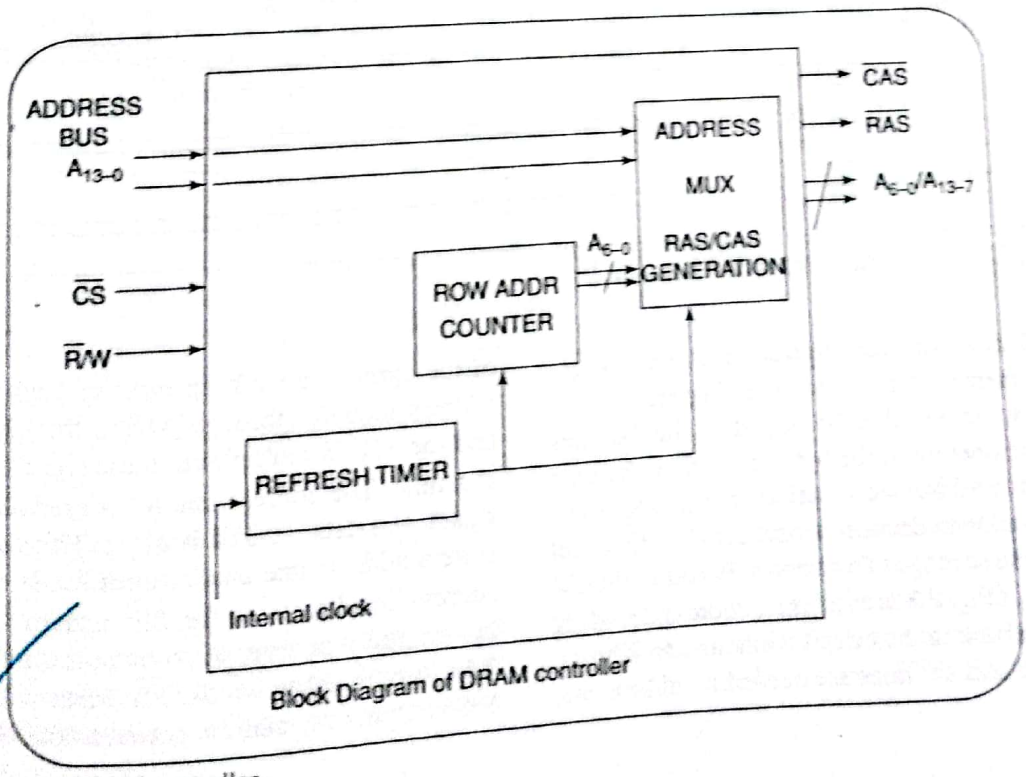


Fig. 8.6 Block diagram of DRAM controller.

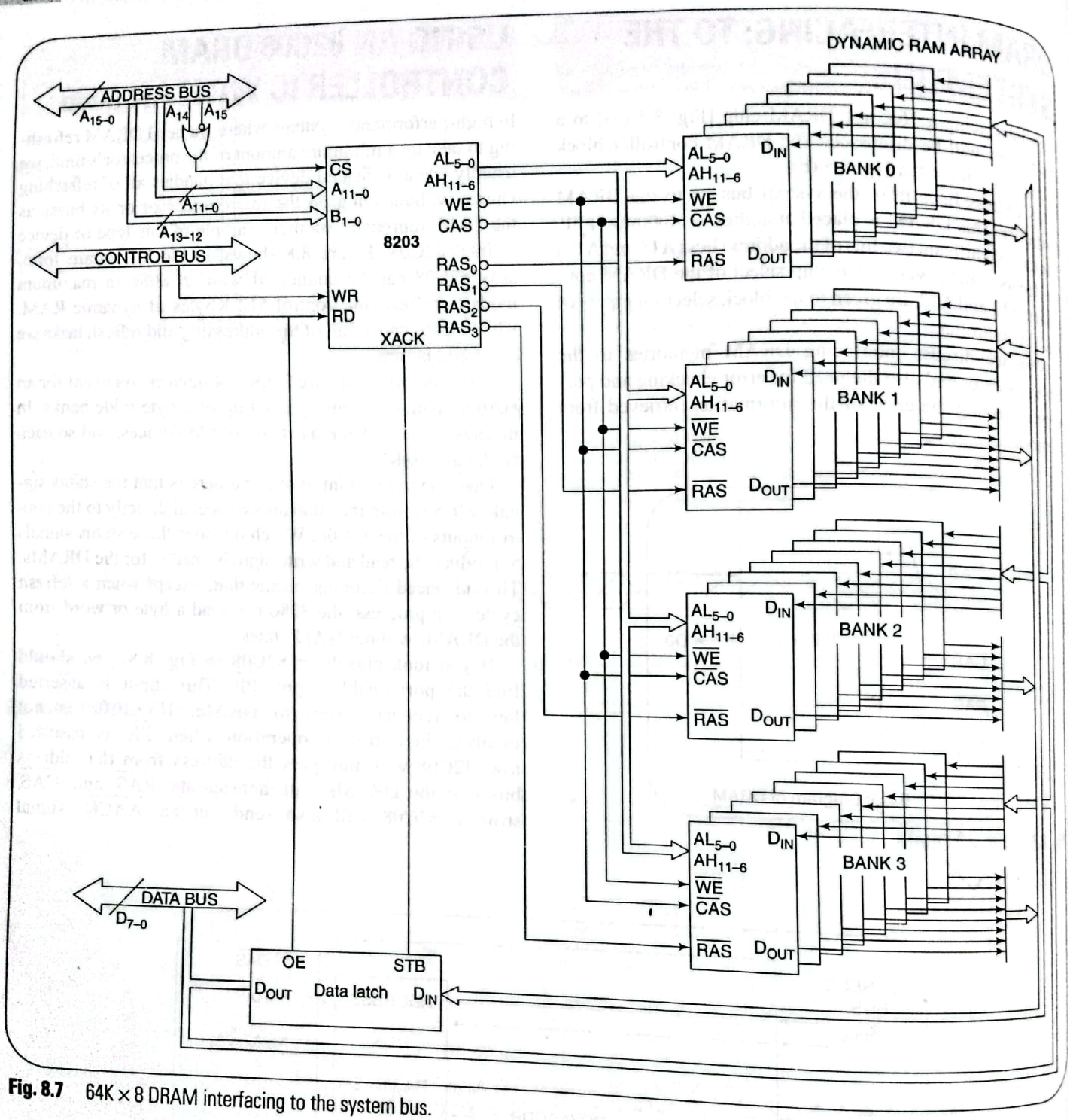


Fig. 8.7 64K × 8 DRAM interfacing to the system bus.

that clocks the 74LS74 flip-flops to transfer A0 and $\overline{BHE}$ signals to the two memory banks. For a read operation, the addressed byte or word will then be output on the data bus to 8086. For a write operation, the byte or word on the data bus will be written to addressed locations in the DRAMs.

The output of an address decoder is connected to $\overline{PE}$ input to assert it for the desired range of addresses. Because DRAM banks in the circuit in Fig. 8.8 are so large, address decoding is very simple. Each bank in the circuit contains 256 Kbytes. Since $256K = 2^{18}$, 18 address lines are needed to address one of the bytes in a bank. In most systems we connect system address lines A1 through A18 to the 82C08 address inputs and the 82C08 multiplexes these signals into DRAMs, nine at a time. The address line A0 is used along with the $\overline{BHE}$ signal to select desired bank(s). This leaves only the A19 system address line unaccounted for. If we connect the A19 address line directly to the $\overline{PE}$ input of 82C08 then $\overline{PE}$ will be asserted whenever 8086 outputs a memory address with A19 low. In other words, $\overline{PE}$ input will be asserted when 8086 outputs any address between 00000H and 7FFFFH.

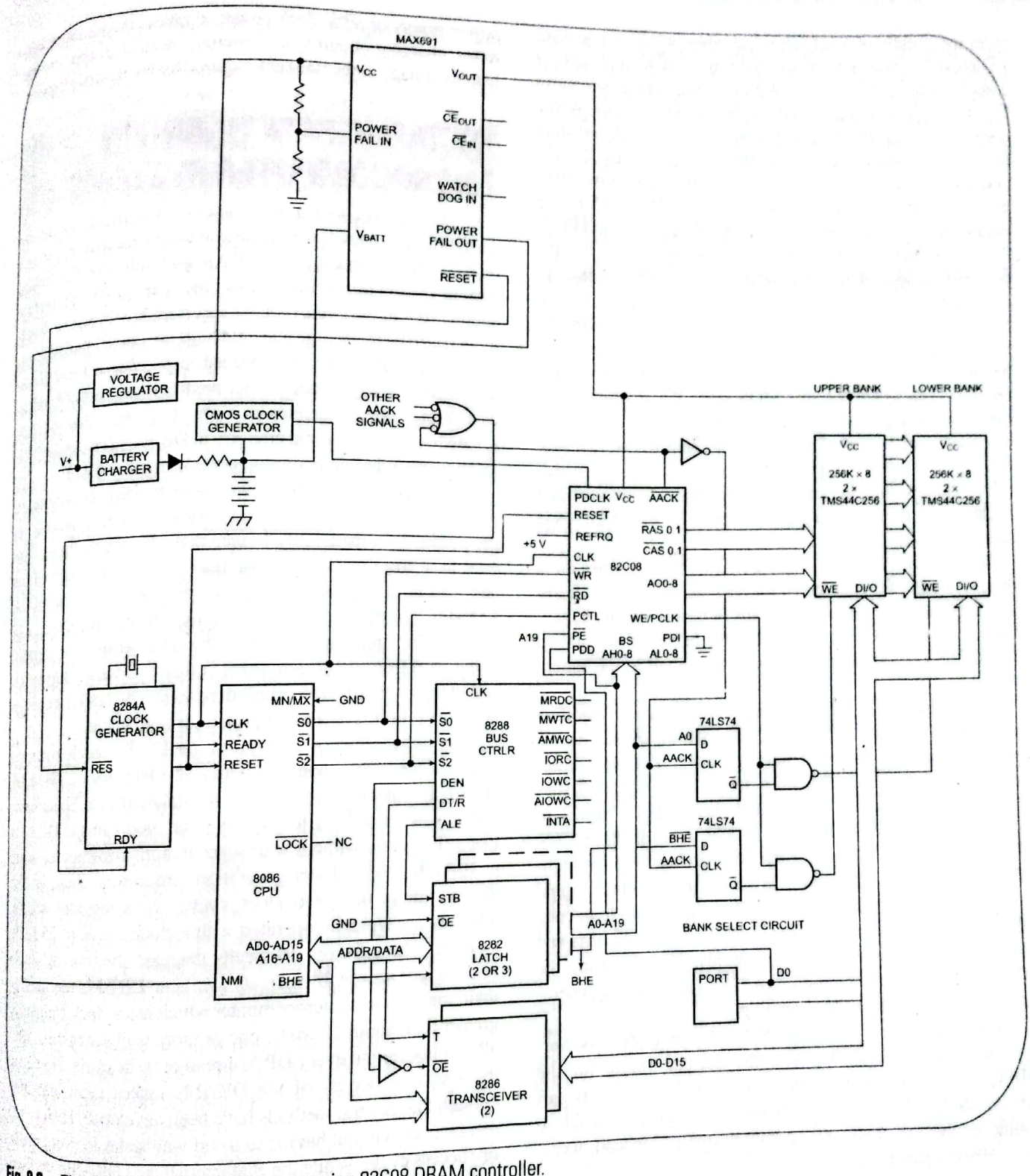


Fig. 8.8 The 8086 Microcomputer system using 82C08 DRAM controller.

NOTE: The status signals from 8086 are decoded in 82C08, so it knows whether an address is intended for memory or an I/O port.

The address decoder here is simply a piece of wire or circuit trace which connects A19 to PE input. This connection puts RAM in the lower half of 8086 address range, which is appropriate, because for an 8086 we need ROMs containing the startup program to be at the top of the address range.

The next point to consider in the system in Fig. 8.8 is how the controller arbitrates the dispute which occurs if the CPU tries to read from or write to the memory while the controller is doing a refresh cycle. If 82C08 in Fig. 8.8 happens to

be in the middle of a refresh cycle when 8086 tries to read a DRAM location, 82C08 will hold its $\overline{AACK}$ high until it completes the refresh cycle. With the connections shown in Fig. 8.8, this will cause 8086 to insert one or more WAIT states while 82C08 finishes its refresh cycle. In this system then, the occasional access conflict is arbitrated by DRAM controller. Inserting a wait state now and then slows 8086 down less than DMA approach used in IBM PC/XT-type computers.

Another interesting feature of the system in Fig. 8.8 is the battery-backup circuitry. We have discussed already the use of an 8086 NMI interrupt procedure to save program data in case of a power failure. In the few milliseconds between the time the ac power goes off and the time the dc power drops below operating levels, an interrupt procedure copies the program data to a block of CMOS static RAM which has a battery-backup power supply. When the system is repowered, the saved data is copied back into the main RAM, and processing takes up where it left off. In larger systems there may not be enough time to copy all the important data to another RAM, so we simply use a battery-backup for the entire RAM array, as shown in Fig. 8.8.

In this circuit we used CMOS DRAMs, because when these devices are not being accessed for reading, writing, or refreshing, they take only a few microwatts of power. During battery backup of the DRAMs they must still be refreshed, and so the 82C08 DRAM controller is also connected to the battery power.

When the power supply voltage drops below a specified level, the PFO pin on the MAXIM 691 supervisor device sends a signal to the NMI input of 8086. The NMI interrupt procedure saves parameters letting the program restart correctly when power returns and then sends a signal to the POWER DOWN DETECT (PDD) input of the 82C08. In response to this signal, the 82C08 switches from the high-frequency system clock to a lower-frequency clock signal from the CMOS crystal oscillator. Reducing clock frequency decreases the amount of current required by the DRAMs and by 82C08 to perform refresh operations. Also, by using the CMOS oscillator, the high-current 8284 system clock generator does not need to be kept running.

When the power returns, the MAX691 generates a power-on-reset signal, $\overline{RESET}$, with the correct timing for the 8086. If a low is output to the PDD input of 82C08 as part of the startup sequence, 82C08 will automatically switch to using the system clock and operate normally for read, write, and refresh operations.

For the battery backup we use a nickel-cadmium alloy or some other type of alloy that can stand the continuous recharging and supply the needed current. The diodes in the circuit prevent the power supply output and the battery from clashing with each other.

In applications where the entire system must be kept running during an ac power outage, we use a *noninterruptible power supply* or *NPS*. Such power supplies contain large batteries, charging circuitry, and circuitry needed to convert the battery voltage to the voltages required by the microcomputer.

DYNAMIC RAM TIMING IN MICROCOMPUTER SYSTEMS

If you take a close look at the read-cycle specifications for the TMS44C256 you will notice see that valid data will be present on the output a time $t_{a(R)}$ after $\overline{RAS}$ goes low. For the fastest current version of the device, this time is about 100 ns. Before another row in the device can be accessed, however, the $\overline{RAS}$ input has to be made high and held high for a time labeled $t_{w(RH)}$. This time of about 80 ns is needed to *precharge* the DRAM so that it is ready to accept the next row address. (Reading data from a storage location in a row discharges that location somewhat and the internal circuitry in DRAM "precharges" the location again before it allows access to another row.)

The precharge time effectively adds to the access time, so the time before a data bit from another row is available on the output is considerably longer than the access time. The total time from the start of one read cycle to the start of the next can be considered as $t_{c(rd)}$. For the fastest version of the TMS44C256 the access time is only 100 ns, but the $t_{c(rd)}$ is 190 ns. For applications where the data words are rapidly being read from random rows, it is this $tc_{(rd)}$ which limits the rate at which words from random rows can be read. Let us see how this time fits in a microprocessor read cycle.

The 8086 requires four clock cycles for each memory access. If the 8086 is operated with a 10-MHz clock (100 ns per clock), the memory access cycle will take 400 ns. This means that if you are willing to pay the price, you can get DRAMs that will operate without wait states in a microcomputer using a 10-MHz 8086. Later-generation processors such as the 80386 require only two clock cycles for a memory access, and they are typically operated with a clock signal of 25 MHz or more. These factors drastically decrease the time available for memory access. If currently available DRAMs are used as main memory in a microcomputer which has a clock frequency greater than about 15 MHz, one or more wait states must usually be inserted in every DRAM read or write cycle. However, the low cost per bit of the DRAMs makes them attractive enough that several methods have been developed. Hence they can be used without having to insert wait states in every memory access cycle. While the characteristics of DRAMs are fresh in your mind, we will introduce here some of these techniques.

PAGE MODE AND STATIC COLUMN MODE DRAM SYSTEMS

Two of the most commonly used techniques to reduce the number of wait states required with DRAMs are the *page*

mode method and the *static column* method. Here's how they work.

In a precharge, time is required each time a new row (page) is accessed in the DRAM. This precharge time is the reason that typical read and write cycle times are so much longer than the access times for DRAMs. If successive data words are read from or written to locations in the same page (row), no precharge time is required. Also, if successive data words are read from the same page, the row address is same, so a new row address does not have to be sent out and strobed in with an $\overline{RAS}$ signal. With a proper DRAM controller, these two factors make it possible to read data from a page or write data to a page without wait states. Some timing diagrams should help you see this.

Figure 8.9 shows the read timing waveforms for a Texas Instruments TMS44C256 DRAM which can be used for page mode access. For the first access in a row (page), the DRAM controller carries out a normal row address-$\overline{RAS}$, column address-$\overline{CAS}$ sequence of signals. If the next address, the controller sends out the same row, and an external comparator will send a signal to the DRAM controller. In response to this "same-row" signal, the DRAM controller will hold the $\overline{RAS}$ low, send out just the column address to the A0—A8 inputs of the DRAMs, and hold pulse $\overline{CAS}$ low. As long as the microprocessor continues to access memory locations in the same page (row), the controller will simply hold the $\overline{RAS}$ low, send out the column part of the addresses to the DRAMs, and hold pulse $\overline{CAS}$ low for each new column address. These accesses within a page are much faster because they need no row address and $\overline{RAS}$ time, and because they need no precharge time.

To determine if a memory access is within the same page, a device such as SN74ALS6310 is connected to the address bus. This device holds the page part of the previous address in a register and compares it to the page part of the new address. If the two address parts are same, the 6310 signals the DRAM controller to do a page mode access as shown in Fig. 8.9. If the previous page address and the current page address are different, the controller will do normal $\overline{RAS}$ and $\overline{CAS}$ access.

Figure 8.9 shows the read timing waveforms for a Texas Instruments TMS44C257 DRAM designed for static column mode operation. During the first access in a row, the DRAM controller carries out a normal row address- $\overline{RAS}$, column address-$\overline{CAS}$ sequence of signals. If the next address the controller sends out is in the same row, an external comparator will signal the DRAM controller. In response to this "same-row" signal, the DRAM controller will hold $\overline{RAS}$ and $\overline{CAS}$ low and send out just the column address to the A0–A8 inputs of DRAMs. As long as the microprocessor continues to access memory locations in the same page (row), the controller will simply hold $\overline{RAS}$ and $\overline{CAS}$ low and send out column part of the addresses to the DRAMs. The static column mode is more difficult to implement than the page mode, but is faster than the page mode since it does not require $\overline{CAS}$ strobes and associated setup and hold times.

In a high-speed microprocessor system, the static column decode technique can reduce the average number of wait states per memory access from 2 or 3 to perhaps 0.8. This is a worthwhile improvement.

ERROR DETECTING AND CORRECTING IN DRAM ARRAYS

Parity Generation/Checking

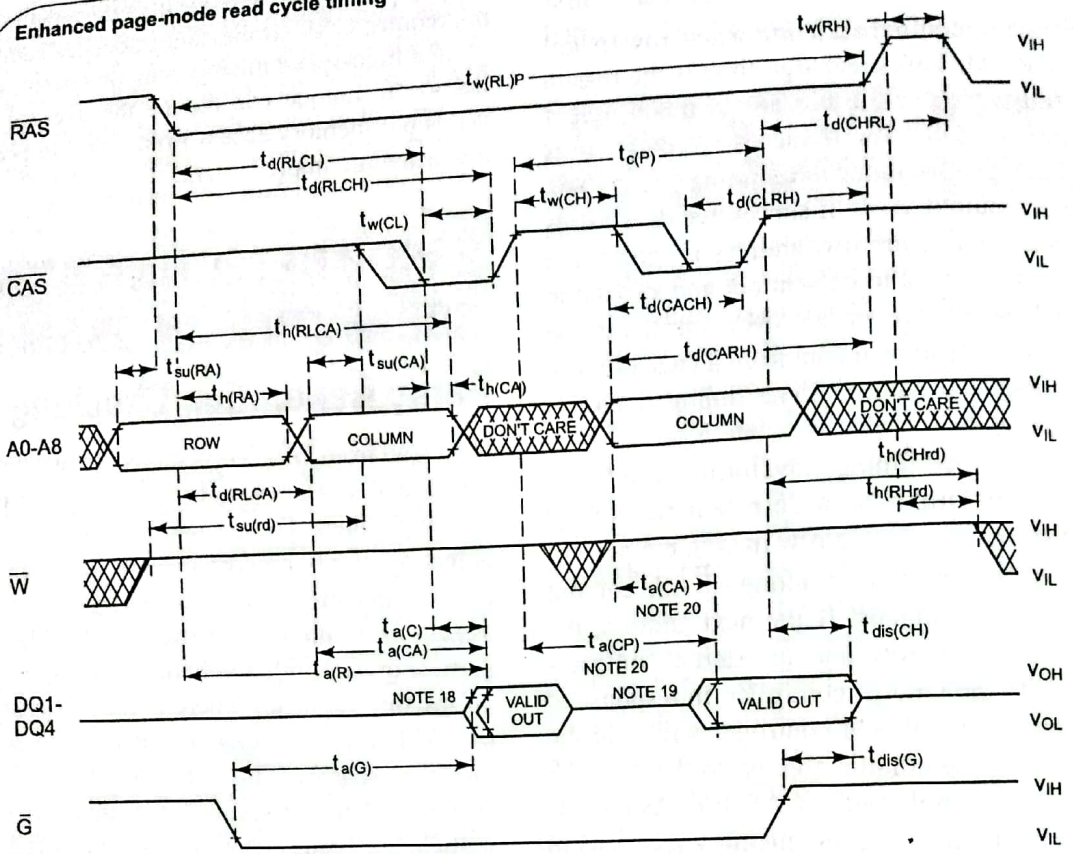
Data read from DRAMs is subject to two types of errors, *hard errors* and *soft errors*. Hard errors occur due to permanent device failures. These may be caused by a manufacturing defect or simply random breakdown in the chip. Soft errors are one-time errors that take place due to a noise pulse in the system or, in the case of dynamic RAMs, perhaps an alpha particle or some other radiation causing the charge to change on the tiny capacitor where a data bit is stored. As the size of RAM array increases, the chance of a hard or a soft error increases sharply. This increases the chance that the entire system will fail. It seems unreasonable that one small alpha particle may cause an entire system to fail. To prevent or at least reduce the chances of this kind of failure, we add circuitry that detects and in some cases corrects errors in the data read out from DRAMs. There are several ways to do this, depending on the amount of detection and correction needed.

The simplest method for detecting an error is with a parity bit. Note that the DRAM memory bank in this circuit is 9 bits wide. Eight of these bits are the data byte being stored, and the ninth bit is a parity bit used to detect errors in the stored data. A 74LS280 parity generator/checker circuit generates a parity bit for each byte and stores it in the ninth location as each byte is written to the memory. When the 9 bits are read out, the overall parity is checked by the parity generator checker circuit. If the parity is not correct, an error signal is sent to the NMI logic to interrupt the processor. When we first turn on the power to an IBM PC or warm boot it by pressing the Ctrl, Alt, and Del keys at the same time, one of the self-tests it performs is to write byte patterns to all RAM locations and check if the byte is read back and the parity of that byte are correct. If any error is found, an error message is displayed on the screen so we don't try to load and run programs in defective RAM.

ERROR DETECTING AND CORRECTING CIRCUITS

One drawback with a simple parity check is that two errors in a data word may cancel each other. A second problem with the simple parity method is that it does not tell you which bit

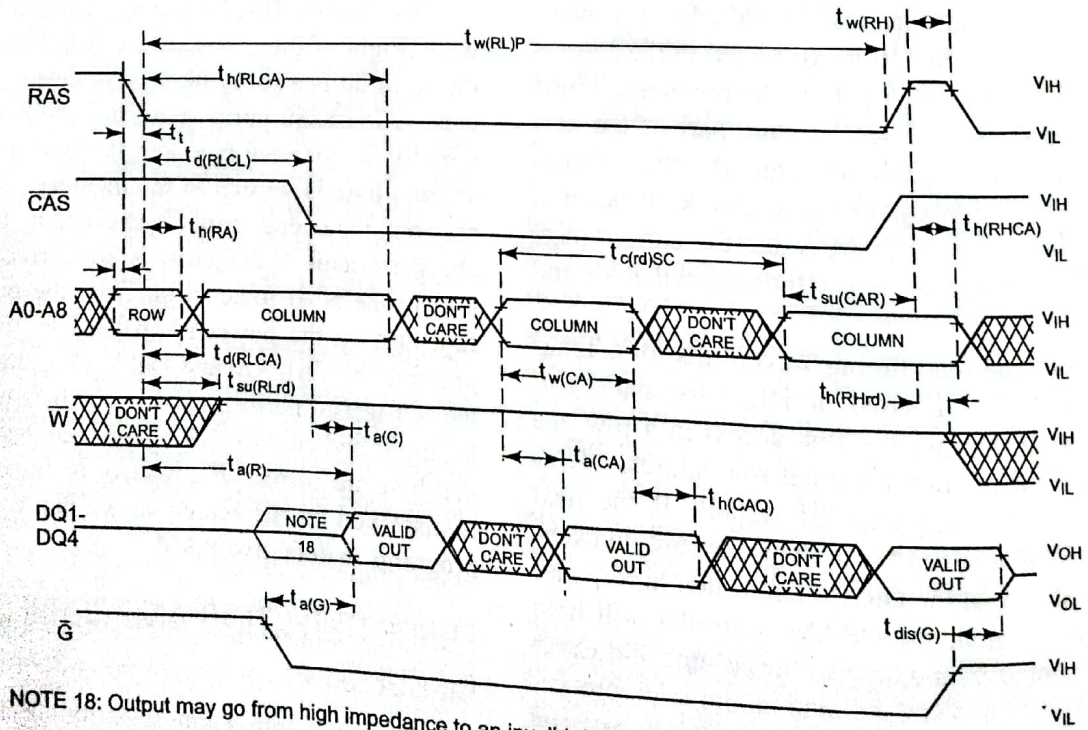
Enhanced page-mode read cycle timing



NOTES: 18. Output may go from high impedance to an invalid data state prior to the specified access time.
19. A write cycle or read-modify-write cycle can be mixed with the read cycles as long as the write and read-modify-write timing specifications are not violated.
20. Access time is $t_{a(CP)}$ or $t_{a(CA)}$ dependent.

(*a*)

Static column decode mode read cycle timing



NOTE 18: Output may go from high impedance to an invalid data state prior to the specified access time.

(*b*)

Fig. 8.9 TMS44C256 DRAM. (a) Page mode read-cycle operation, (b) Static column read-cycle operation. (Texas Instrument Inc.)

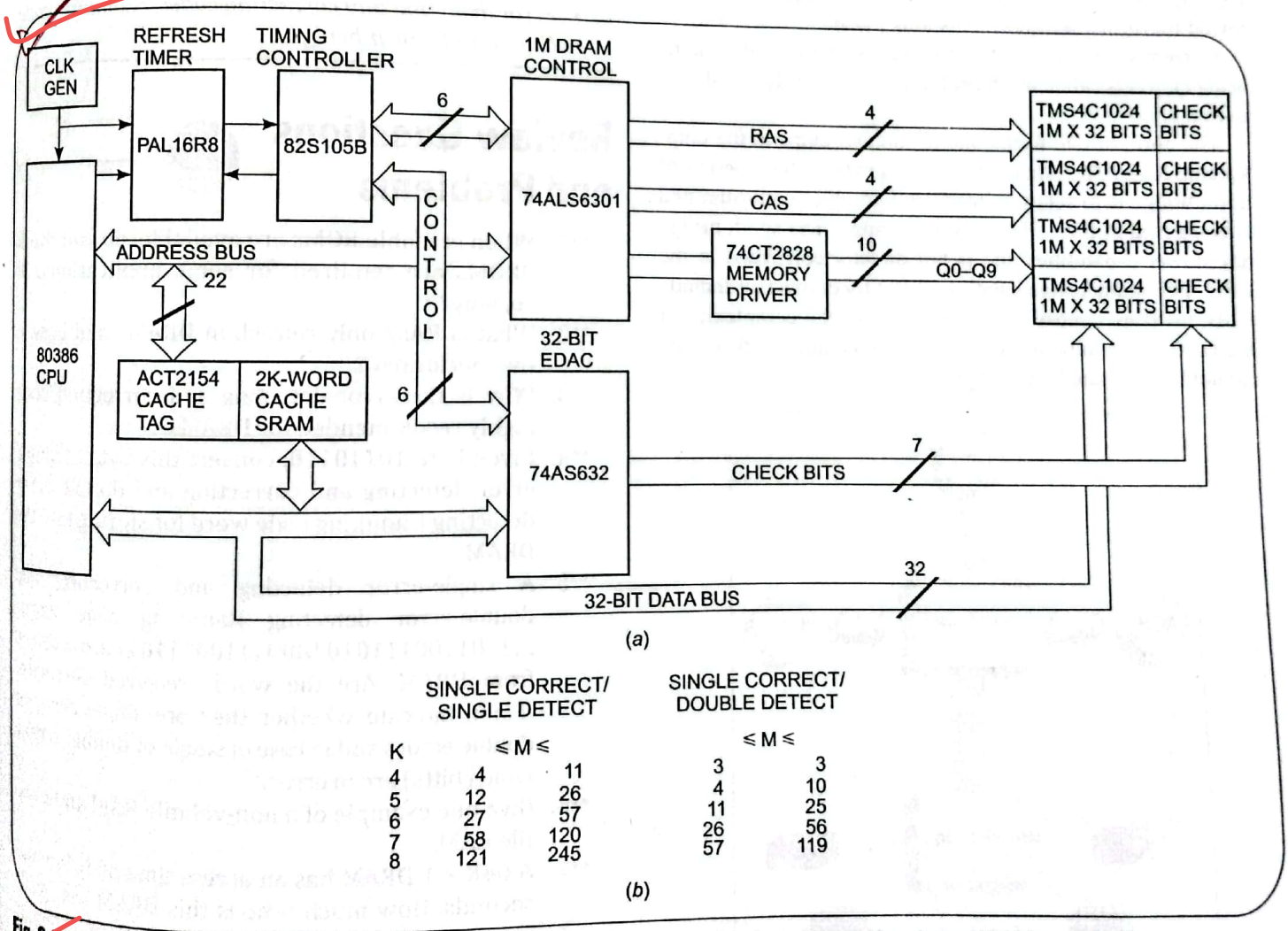
in a word is wrong so that you can correct the error. More complex error detecting/correcting codes (ECCs), often called *Hamming codes* (after the man who did some of the original work in this area), allow you to detect multiple-bit errors in a word and to correct at least one bit error.

Figure 8.10 shows in block diagram form how a T1 74AS632 error-detecting and correcting (EDAC) device can be connected in the data path between a 32-bit microprocessor and 16-Mbyte DRAM main memory. The EDAC is connected in parallel with the DRAM refresh controller and in series with the SRAM cache. Let us see the EDAC device works.

When a data word is sent from the microprocessor to the memory, it also goes to the EDAC. As the data word is read in by the EDAC, several *encoding* or *check* bits are generated and written in memory along with the data word. As shown in Fig. 8.10, the number of encoding bits, K, required is determined by the size of the data word, M, and the degree of detection/correction needed. The total number of bits required for a data word N is equal to M + K. For example, 5 encoding bits are needed to detect and correct a single-bit error in a 16-bit data word, so a total of 21 bits have to be stored for each 16-bit word. To detect/correct a 1-bit error and detect 2 wrong bits in a 32-bit word requires 7 encoding bits, or a total of 39 bits. The encoding bits, incidentally, are not just tacked on to one end of the data word as a parity bit is. They are spread out in the data word.

When the processor reads a data word from the memory, the data word and the check bits from memory go to the EDAC. The EDAC calculates the check bits for the data word read out from the memory and XORs these check bits with the check bits that were stored in memory with the data word. The result of this XOR operation is called *syndrome word*. The syndrome word is decoded to find out if the data word has no errors, has a single-bit error, or has multiple-bit errors.

If the data word contains no errors, the 74AS632 EDAC will simply output the data word to the processor on the data bus. If the data word contains a single-bit error, the EDAC



(*a*)

K	SINGLE CORRECT/ SINGLE DETECT ≤ M ≤		SINGLE CORRECT/ DOUBLE DETECT ≤ M ≤	
4	4	11	3	3
5	12	26	4	10
6	27	57	11	25
7	58	120	26	56
8	121	245	57	119

(*b*)

Fig. 8.10 (a) Block diagram showing how error detecting and correcting circuitry is connected in a large DRAM system. (Texas Instruments Inc.) (b) Hamming-code data bits and encoding bits and number of encoding bits required for desired degree of detection/correction.

device uses the syndrome word to find out which bit is incorrect and simply inverts that bit to correct the bit. The EDAC then outputs the corrected data word to the processor on the data bus. If the data word contains multiple-bit errors, the EDAC device asserts a signal that is usually connected to an interrupt input on the processor. In case of a multiple-bit error, the programmer must decide what action to take and write the appropriate interrupt-service procedure.

74AS632 EDAC in Fig. 8.10 can also work with the 74ALS6301 DRAM controller to remove errors in stored data words during refresh operations as well as normal read operations. This is called *scrubbing*. Correcting errors during each refresh operation decreases the chance of multiple-bit errors accumulating between read operations.

For more information on DRAM error detecting/correcting, consult the data sheets for error detecting/correcting devices such as the Intel 8206, the Texas Instruments 74AS632, or the National DP8402A.

Memory interfacing techniques described above are to give a know-how of how memories can be connected to the CPU through the system bus. These techniques were followed till further advancements took place.

Currently interfacing of memories and peripherals is done using chip sets called north bridge and south bridge as shown in Fig. 8.11.

The north bridge forms one of the two chips in the chipset core logic. The other is the *south bridge*. The function of north bridge is to act as memory and graphics controller and is connected to the CPU and south bridge. The south bridge has slower capabilities and is not directly connected to the CPU. The south bridge handles all I/O functions that include USB, interrupt controller, etc. Because of the complexity of hardware, all components of north bridge and south bridge cannot be integrated on a single chip.

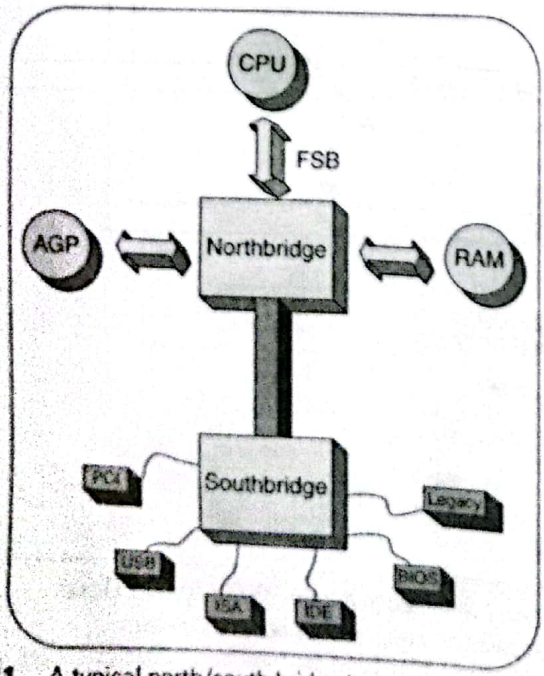


Fig. 8.11 A typical north/south bridge layout.

With new developments in technology functions, both north and south bridges are being gradually integrated into the CPU chip itself and examples of such CPUs are Intel *Sandy Bridge* and *AMD Fusion*. These were introduced in 2010/2011. Hence, memory interfacing has become less complex.

Checklist of Important Terms and Concepts in This Chapter

- *ROM, PROM, EPROM, EEPROM, FLASH ROM, SRAM, DRAM*
- *Fast page mode DRAM<NVRAM, DDR Family, DRDRAM*
- *Synchronous memories*
- *Memory device configurations*
- *Memory interfacing*
- *DRAM controllers 8203, 82C08*
- *Hard and soft errors*
- *Error detecting and correcting codes, Hamming code*
- *North and south bridges*

Review Questions and Problems

**1. When erasable ROMs are available, do you think PROMs are required for some applications? If so, why?

**2. What is RAS/ only refresh in DRAMs and how is that performed?

*3. Why is the error-detecting and correcting code highly recommended for DRAMs?

**4. Given byte 10110110, convert this byte to single error detecting and correcting and double error detecting Hamming code word for storing into the DRAM.

***5. A single-error detecting and correcting and double-error detecting Hamming code words a)1001100111010 b)0011100111011 are received from DRAM. Are the words received correctly? If not, indicate whether they are single error or double errors and in case of single or double errors, which bit(s) are in error?

**6. Give one example of a non-volatile RAM and volatile RAM.

***7. A 64K × 1 DRAM has an access time of 100 nanoseconds. How much time is this DRAM not available to CPU if the DRAM refreshes in a) burst mode, or b) RAS/ only refresh mode.

**8. List the major tasks that must be done to support dynamic RAM in a microcomputer system.

*9. How does a dynamic RAM controller in a system such as that in Fig. 8.8 arbitrate the dispute that occurs when the CPU attempts to read from or write to a bank of dynamic RAMs while the controller is doing a refresh cycle?

***10. (a) What timing parameter limits the rate at which data words can be read from random rows (pages) in a DRAM?

(b) Explain how page mode operation of a bank of DRAMs makes it possible for a microprocessor to read data words without wait states.

(c) What is the main difference between page mode operation and static column mode operation of a bank of DRAMs?

*11. Describe how parity is used to check for RAM data errors in microcomputers such as the IBM PC. What is a major shortcoming of the parity method of error detection?

*12. When using Hamming code error detection/correction scheme for DRAMs, how many encoding bits must be added to detect and correct a single-bit error in a 64-bit data word?