MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

Order this document
by AN1125/D

AN1125

DRAM Interface to the MC88200 M Bus

This document describes the design and operation of a 25 to 33.3 MHz dynamic random-access memory (DRAM) interface to the M Bus of the Motorola MC88200 Cache/Memory Management (CMMU) Device. The memory interface is divided into two sub-systems: a CPU sub-system and a memory sub-system. The CPU sub-system includes one MC88100 CPU and two MC88200 CMMUs. The memory sub-system contains a DRAM array (which consists of two non-interleaved, 32-bit wide banks of single in line memory module (SIMM) devices with byte parity) and a DRAM controller. The DRAM controller is implemented using programmable array logic (PAL), a Xilinx (a gate array), and discrete hardware. The memory sub-system is capable of parity generation and checking, automatic refresh, and burst mode accesses.

The memory interface system described in this document has been extracted from a larger application; therefore, some parts of the system are implemented in peripheral devices that are not discussed in this document. This design has been successfully implemented; however, it is not the only possible configuration of a memory interface to the MC88200.

The audience of this document is intended to be hardware systems designers with some knowledge of the M88000 family and some familiarity with DRAMs.

NOTE

In this application note, assertion and negation are used to specify forcing a signal to a particular state. In particular, the terms assertion and assert refer to a signal that is active or true; negation and negate indicate a signal that is inactive or false. These terms are used independently of the voltage level (high or low) that they represent. Throughout this document, active high signals are denoted by the signal name (i.e., SIGNAL), and active low signals are denoted by the signal name with an overbar (i.e., SIGNAL).

DESIGN OBJECTIVES

There were three objectives for this design. The first was to make a high performance system while limiting the cost of hardware and the board space used. The next was to protect data integrity using parity. The final objective was to make the design flexible.

The first objective was not quantifiable, but provided a general guideline for the design. Some performance enhancements such as interleaved memory and pipelined data paths could have been added to the design. However, each of these enhancements would have added a considerable amount to the total system cost and the board space used.

Parity checking was included in the design to protect data integrity on read accesses. Parity checking can be performed in series or in parallel. With series parity checking, the processor read is delayed until after parity evaluation has been performed on the data. In parallel parity checking, the processor reads the data first and is notified after the read whether the data is invalid; thus, parallel parity detection minimizes the read cycle latency relative to sequential parity checking. However, parallel parity checking is somewhat more complicated than sequential checking because with parallel checking the system must interrupt the processor after an error is detected. Also, with parallel parity checking, the memory location of the parity error must be stored since the CPU may already have begun executing another instruction when the interrupt is received. Parallel parity checking was used in this memory interface system. The higher performance of parallel checking relative to serial checking outweighed the cost of the few extra components needed for parallel checking.

The last objective of the design was to make this system flexible. By using programmable logic, some areas of the system have been designed to allow flexibility. For example, this design supports both page and nibble-mode DRAMs to allow the end user the choice based on price and availability. Also, included in this system is the flexibility to add up to two more banks of DRAMs with little change to the control logic.

OVERVIEW

The DRAM interface system consists of a CPU sub-system and a memory sub-system (see Figure 1). An overview of these two sub-systems is given in the following paragraphs.

CPU SUB-SYSTEM

The CPU sub-system contains one MC88100 CPU, two MC88200 CMMUs, system reset logic, and an MC88914 low skew CMOS clock driver. The MC88100 CPU is the engine for the system. The MC88200 devices provide high speed caching for both the instruction and data busses, facilitating no-wait-state memory operations for the MC88100. Each MC88200 device contains 16 KBytes of 4-way set associative cache as well as a two-level, demand-paged memory management unit and support for shared memory multiprocessing.

The MC88914 low skew CMOS clock driver is used to synchronize the clock inputs for the CPU and two CMMUs and to generate the system clock. The MC88914 is supplied with a 50 or 66.6 MHz oscillator. The clock driver divides the frequency in half, and generates multiple, synchronized 25 or 33.3 MHz frequency clock signals. The outputs of the MC88914 are guaranteed to provide duty cycles and rise and fall times which meet the specifications of the CPU and CMMUs.

The M Bus signals interface the MC88200s to the memory system. The M Bus is a shared, multiplexed bus that accesses memory and memory-mapped peripheral devices.

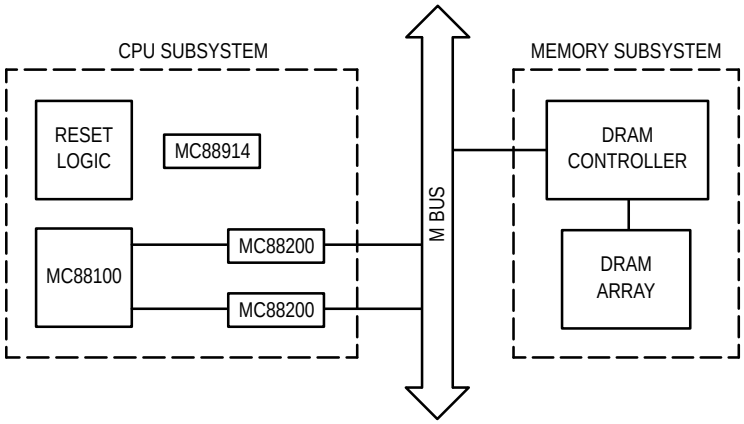

Figure 1. Functional Block Diagram of the DRAM Interface

The M Bus includes a bus arbitration scheme that grants bus ownership to a single M Bus master. Table 1 lists the M Bus signals.

Table 1. M Bus Signals

Signal Name	Mnemonic	Type	Active
Bus Request	BR	Output	High
Bus Grant	BG	Output	High
Bus Acknowledge	BA	Output	High
Address/Data	AD31 – AD0	I/O	High
Address/DataParity	ADP31 – ADP0	I/O	High
Local Status	ST3 – ST0	I/O	High
System Status	SS3 – SS1	Input	Low

The four system status signals ($\overline{SS3}$ – $\overline{SS1}$) provide the M bus master with a status reply which is generated by M bus slaves to reflect the status of the M bus. Table 2 lists the values of these signals. The local system status signals (ST3 – ST0) of the two CMMUs and the OK and WAIT signals generated by the memory controller are wire-ORed together to produce the system status signals.

Table 2. M Bus System Status Signals

Status	SS3	SS2	SS1	SS0
Error	0	X	X	X
Retry	1	0	X	X
Wait	1	1	0	X
End of Data (Data Phase)	1	1	1	0
OK (Phase Completed)	1	1	1	1

The M bus has control signals (C6 – C0) which specify read and write operations, establish exclusive resource usage (locks), inhibit external caching, signal bus errors, and provide initialization. C0 Indicates whether an MC88200 read or write cycle is in the address phase or data phase: when C0 is asserted, the MC88200 is in the address phase, and when C0 is negated, the MC88200 is in the data phase. Control signals C6 – C1 have different meanings during the address and data phases of a cycle. A description of the control signals and their significance during address and data phases is provided in Table 3.

Table 3. MC88200 M Bus Control Signals

Signal	Description
Address Phase (C0 Asserted)	
C1	Intent to Modify. Asserted if the transaction is a memory write or a read with intent to modify. This signal is used for data cache coherency control.
C2	Asserted for memory read. Negated for memory write.
C3	Lock. Asserted to lock out all other M bus devices.
C4	Cache Inhibit. Asserted if the MC88200 cannot cache the data transferred during the current memory access.
C5	Global. Asserted if the address's memory location may be shared by more than one MC88200.
C6	Reserved. Driven as zero.
Data Phase (C0 Negated)	
C1	Last data transfer. Asserted if the current data phase completes the transfer.
C2	Asserted for memory read. Negated for memory write.
C3	M Bus Byte Enable 3. Asserted if AD31 – AD24 contain significant data.
C4	M Bus Byte Enable 3. Asserted if AD23 – AD16 contain significant data.
C5	M Bus Byte Enable 3. Asserted if AD15 – AD8 contain significant data.
C6	M Bus Byte Enable 3. Asserted if AD7 – AD0 contain significant data.

MEMORY SUB-SYSTEM

The memory sub-system contains address latches, a DRAM array, data latches, parity generators and checkers, refresh logic, main memory control logic, and row address strobe (RAS) and column address strobe (CAS) generation for the DRAM. The DRAM addresses are latched because addresses on the M bus are not held valid across clock edges. The outputs to the latches are always enabled in order to minimize access times.

When the MC88100 tries to access data or an instruction that is not in one of the CMMUs (i.e., the CPU misses in a CMMU), a burst read cycle to memory is performed to fill an entire cache line. The MC88200 makes an initial access to external memory to access the first word in the cache line

and then fills the rest of the corresponding cache line (four 32-bit words total). Because the CMMU performs burst accesses, the DRAMs used in this system support fast access modes (page- or nibble-mode).

The DRAM array is divided into two banks of SIMMs, with four SIMM devices in each bank; thus, each bank has a 32-bit wide data bus with byte parity. Depending on whether one or both banks are filled, this arrangement allows for memory configurations of 4 and 8 Mbytes with 1 Mbit DRAMs or 16 and 32 Mbytes with 4 Mbit DRAMs. Table 4 presents the four different configuration options.

Table 4. Memory Sub-System Signal Descriptions

Signal	Description
MA10 – MA0 (4 Mbit DRAMs) or MA9 – MA0 (1 Mbit DRAMs)	Multiplexed address lines to the DRAM array.
XA10 – XA0 (4 Mbit DRAMs) or XA9 – XA0 (1 Mbit DRAMs)	Buffered multiplexed address lines for DRAM bank0.
YA10 – YA0 (4 Mbit DRAMs) or YA9 – YA0 (1 Mbit DRAMs)	Buffered multiplexed address lines for DRAM bank1.
MD31 – MD0	Data lines to and from the DRAM array.

OVERVIEW

Figure 2 shows a block diagram of the memory sub-system. The memory sub-system consists of five major blocks: address multiplexing logic, DRAM array, data latching, parity generation and checking logic, and DRAM control logic. Each of these blocks is discussed in detail in the following paragraphs.

The Control Bus from the M Bus to the DRAM control logic consists of the following M bus signals (see Table 1): BR, BG, BA, and $\overline{SS3}$ – $\overline{SS0}$. The other signals in Figure 2 are listed in Table 5. Note that 4 Mbit DRAMs have eleven address lines while 1 Mbit DRAMs have ten address lines.

READ AND WRITE CYCLES

Figure 3 shows the basic read and write burst timings for the memory sub-system.

To initiate a read or write access, an MC88200 must become the bus master. For an MC88200 to become bus master, the following actions are performed: 1) the MC88200 requests M Bus mastership by asserting the BR signal, 2) the arbitration logic grants the bus by asserting the BG signal, and 3) the MC88200 then acknowledges the transfer with the BA signal.

The MC88200 has a minimum burst access time of 2:1:1:1. The memory interface system adds two wait states to each access, making the system a 4:3:3:3 system for burst read accesses. The first access is measured from the time when the CPU receives a bus acknowledge (BA) until a valid word of data is presented on the AD lines (signaled by an OK response on the SS3 – SS0 lines). This takes four clocks. After the first valid word of data is received, it takes three clocks to access each subsequent word of data. For burst writes, the memory system adds an additional wait state to the initial access making the system a 5:3:3:3 system for burst write accesses.

To maintain the 4:3:3:3 burst read accesses and 5:3:3:3 burst write accesses, 100 ns DRAMs can be used at 25 MHz, but 80 ns DRAMs are needed at 33 MHz. A more detailed explanation of the data path timing is presented in **ADDRESS MULTIPLEXING**.

BURST CYCLE CONTROL SIGNALS

Figure 4 shows the operation of the signals that control burst transactions.

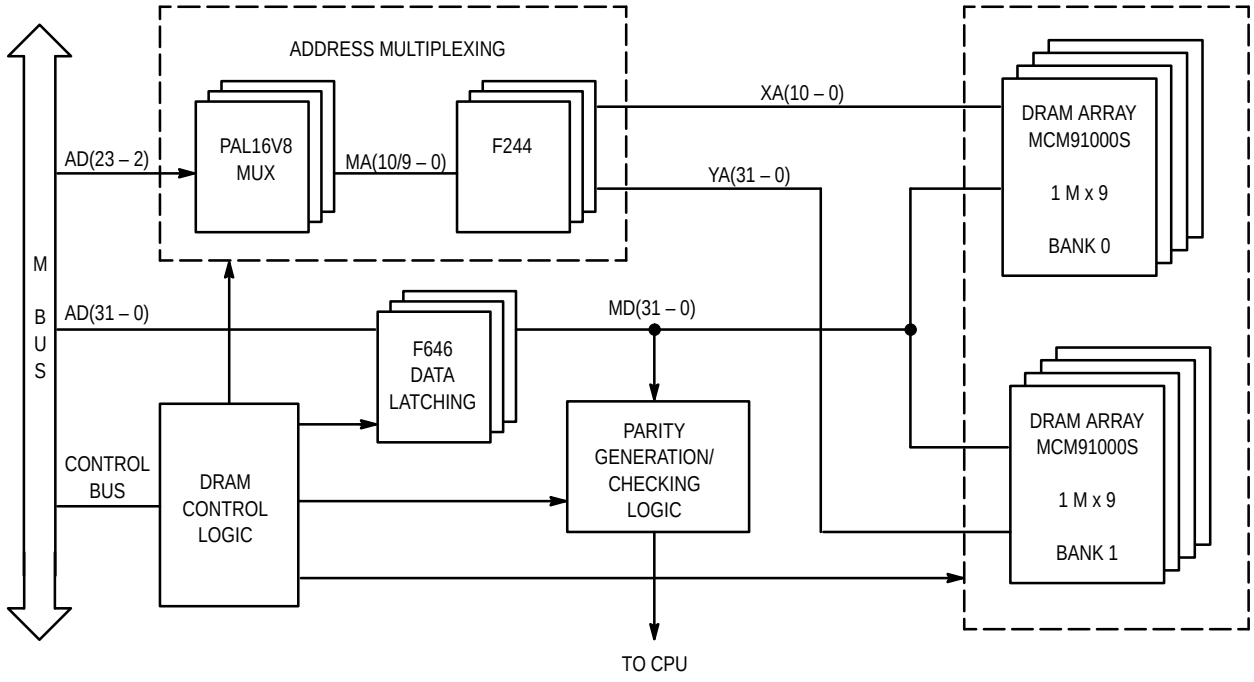

Figure 2. Address/Data Path to DRAM System

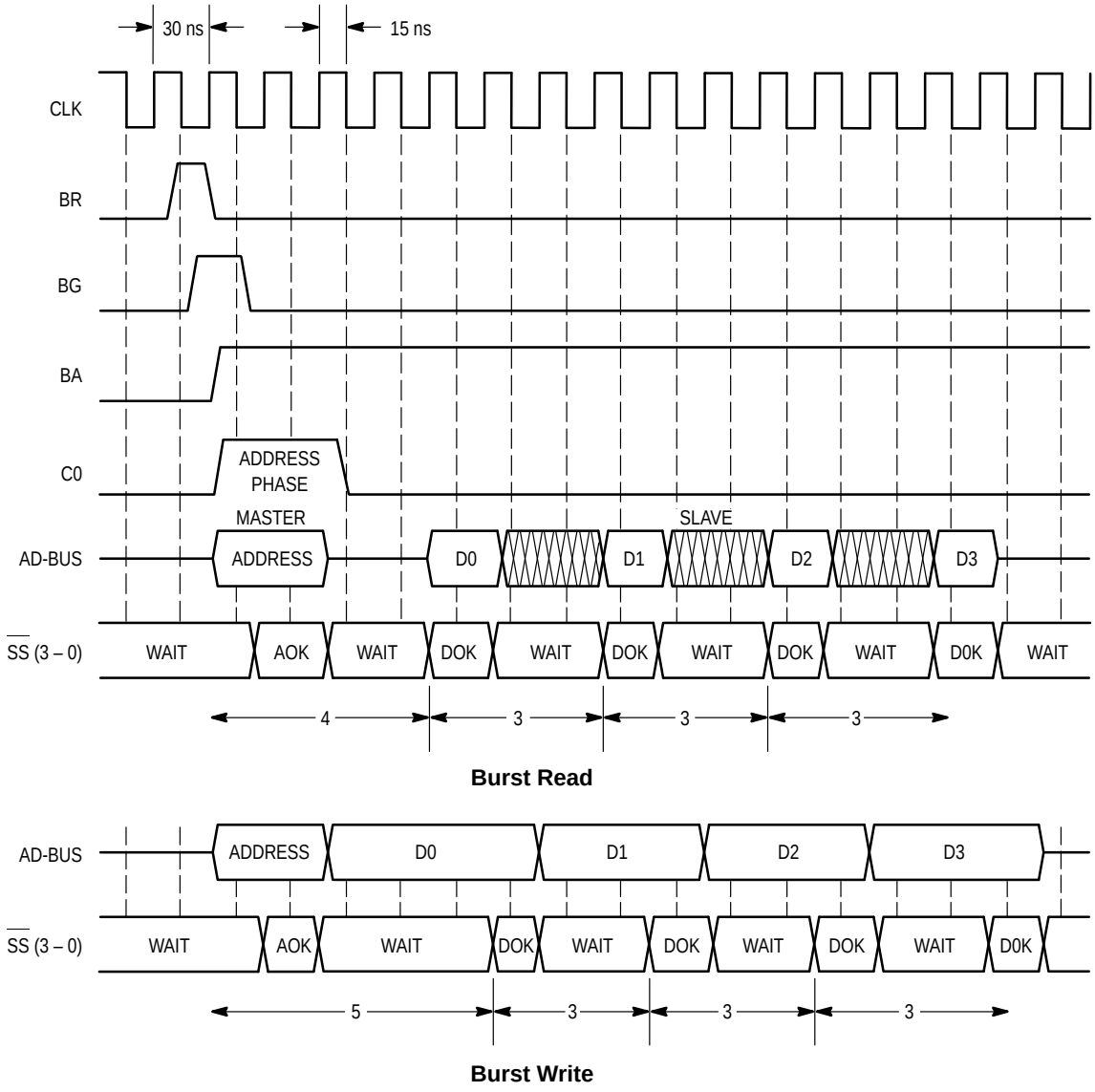

Figure 3. Burst Read and Write Cycle (33.33 MHz) Timings

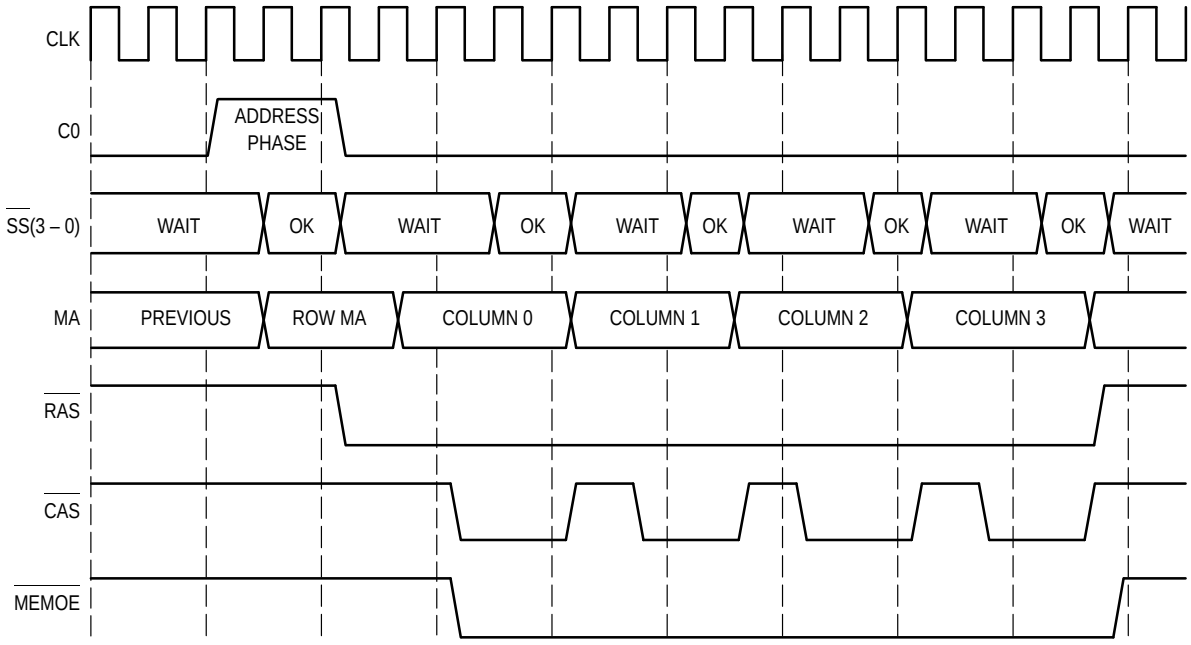

Figure 4. Burst Read Cycle Control Signals Timing

The address on the M Bus is multiplexed to provide the row and column addresses for the DRAM array and is placed on the multiplexed address lines (MA10 – MA0) (see **ADDRESS MULTIPLEXING**). The DRAM control logic starts the data phase of the cycle by asserting the row address strobe (RAS) signal at the end of the address phase. The RAS signal remains asserted, and the column address strobe (CAS) signal toggles low then high for each column address to indicate that a valid address is on the multiplexed address lines. The address is valid before the falling edge of CAS to comply with the specifications of the DRAMs. The memory output enable (MEMOE) signal, generated by the DRAM control logic, is asserted on the same rising clock edge as the first CAS assertion and stays asserted for the entire period of the burst access. MEMOE is the output enable for the 74F646 latches. These latches drive data between the DRAMs and the M Bus.

ADDRESS MULTIPLEXING

Figure 5 shows a block diagram of the address multiplexing hardware used to generate row and column addresses for the DRAM array.

Because the address and data from the MC88200 are time multiplexed on the M bus, the address must be latched during the address phase. Therefore, the multiplexing hardware must be able to latch addresses from the M bus as well as multiplex the addresses for the DRAMs.

In this system, to minimize the board space needed, PALs were used instead of latches and multiplexers. Three 15 ns PAL16V8s latch the address from the M Bus address/data lines (AD23 – AD2) during the address phase (C0 asserted) and then multiplex the address into row and column addresses. These row and column addresses are placed on the multiplexed address lines (MA10 – MA0). Address lines AD1 – AD0 from the M bus are not used in the multiplexing. Instead, control signals C3 – C6 from the MC88200 are used to indicate which byte or bytes are to be accessed. This implementation supports both 1-Mbit and 4-Mbit DRAMs. Because 1-Mbit DRAMs use only 10 address lines, MA10 is ignored when 1-Mbit DRAMs are used. Therefore, the multiplexed address lines for the DRAMs are referred to as MA10/9 – MA0 for the rest of this document.

The addresses on MA10/9 – MA0 are passed through 74F244 buffers (whose purpose is explained in the following paragraphs). The outputs of the buffers (address lines XA10/9 – XA0 and YA10/9 – YA0) are always enabled, and each set of address lines contain the same data as MA10/9 – MA0. Lines XA10/9 – XA0 connect to DRAM bank0 and lines YA10/9 – YA0 connect to bank1. The buffer outputs are passed through series terminating resistors to reduce the amount of ringing on the address lines that connect to the DRAMs.

Each bank of DRAMs has an input capacitive load of 216 pF (54 pF per SIMM) on the address lines, but the PAL16V8s have a drive capability of only 50 pF on each line. Because the PALs do not supply enough capacitive drive to support the DRAMs without much derating, the output from the PALs is passed through 74F244 buffers. The capacitive drive capability of the 74F244s is also specified to 50 pF; however, the 74F244s provide larger sink and source currents than the PALs, thus reducing the derating caused by the capacitive loading of the DRAMs.

Even with the buffers, some derating is still required since each buffer output must drive a capacitive load of 216 pF (four SIMMS) which is 166 pF over the 74F244's specified capability. According to the *Signetics FAST Logic Data Handbook*, 1989, for the 'F00 family of TTL devices, 200 pF of load capacitance changes the propagation delay of the 74F244 to approximately 8 ns from the 3 ns specified for a 50 pF load; thus, the 74F244 buffers are derated 5 ns by the capacitive load.

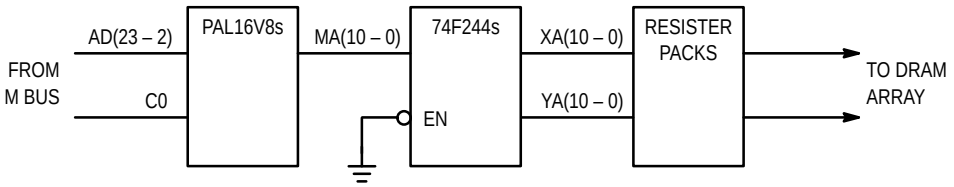

Figure 5. Address Multiplexing

Figure 6 shows the critical path for a DRAM access. The critical path is the signal route starting from the time the control signals are valid to when the DRAMs receive the CAS signals, and ending when the appropriate data reaches the CPU via the M bus. For the critical path, there is a maximum 9.5 ns derate delay allowed for capacitive loading and clock skew.

Equation 1 shows how the maximum derate delay is calculated (all times are in nanoseconds). This equation takes into account all of the critical delays through the data path, given a two wait state memory access. The terms used in equation 1 are defined as follows:

Derate Delay — The amount of time allowed in this system for the delay caused by capacitive loading and clock skew.

WAIT — Two clock cycles at 33.3 MHz. This is because of the two wait states added to each access by the system (see **READ AND WRITE CYCLES**).

CAS — The time required for the PAL16V8 to activate the CAS signal.

F244 — The maximum propagation delay through the CAS buffers.

DRAM — The time required to access the DRAMs starting when CAS goes low.

F646 — The maximum propagation delay through the latching transceivers.

88K Data Setup — The amount of time required for the AD Bus data to be valid before the rising edge of the clock.

(1) Derate Delay <= WAIT – (CAS + F244 + DRAM + F646 + 88k Data Setup)

Derate Delay <=60 – (8 + 6.5 + 20 + 10 + 6)

Derate Delay <= 9.5 ns

The worst case output to output clock skew from the 88914 is 01.0 ns per clock; thus, the derate delay allowed for the 74F244 buffers is 8.5 ns (i.e., 9.5 ns – 1 ns). This is enough time since the 74F244 buffers are derated only 5 ns, as stated previously.

DRAM ARRAY

The DRAM array is composed of two banks of SIMMs with four SIMM devices in each bank (see Figure 7). Each SIMM device contains nine DRAMs; thus, each bank has a 32-bit wide data bus and one bit of parity per byte.

The design of the memory system allows the user to configure the board for page- or nibble-mode DRAMs. Although this option increases the complexity of the hardware, it gives the user flexibility to choose DRAMs based on availability and price.

Page- or nibble-mode is selected by a jumper connected to the PAGE input of the PAGE/NIB PAL. When the PAGE input is asserted (no jumper attached), page mode is selected. The multiplexed nibble (MNIB) output is connected to MA9 in the DRAM array. The buffered MNIB signals, XNIB and YNIB, go to address bit XA9 of bank0 and address bit YA9 of bank 1, respectively, and automatically cycle through four words of data. If the PAGE input is negated (jumper attached), nibble mode is selected and the MNIB Output acts any other address output.

The data on the data in (D_{in}) pins of the DRAMs are latched by the DRAMs during write cycles as indicated by the assertion of the write enable signals, XWE (for bank 0) and YWE (for bank 1). The data out (D_{out}) pins are driven by the DRAMs when CAS is asserted during read cycles (indicated when XWE and YWE are negated).

Flexibility has also been introduced into the design by allowing the user to configure the board for either 1- or 4-Mbit DRAMs. The 1/4-Mbit jumper is connected to a PAL which decodes the lower four bits of the multiplexed address. This PAL decodes the addresses differently depending on whether the jumpers set to 1 Mbit or 4 Mbit.

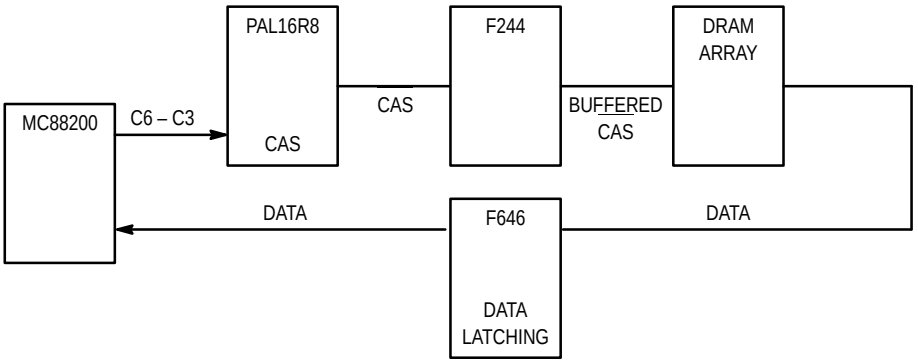

Figure 6. Data Path Flow

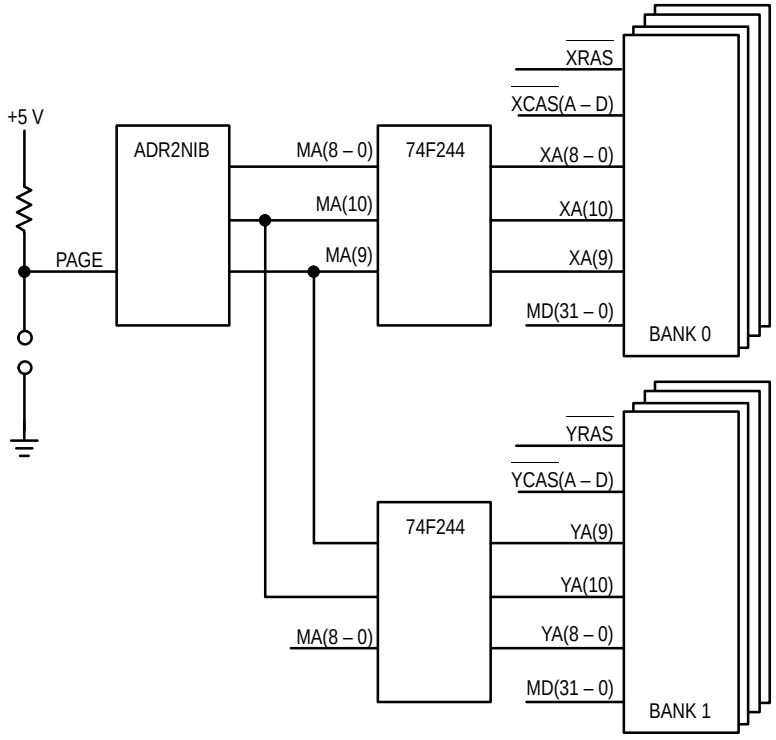

Figure 7. DRAM Array Block Diagram

DATA LATCHING

Because the M bus is not held valid by the CMMUs across clock edges, during a CPU write the data must be latched and held valid while C0 is negated (data phase) to meet the setup times for the DRAM array. On a CPU read, the data does not need to be latched because the DRAM holds the data valid across clock edges.

Four 74F646 registered transceivers (see Figure 8), latch the data going from the M bus to the DRAM array (i.e., in the B-to-A direction, from AD31 – AD0 to MD31 – MD0), but are transparent to data going in the other direction (from A-to-B). The clock for data going in the A-to-B direction is grounded since that data is not latched; however, the data flow from B-to-A is clocked by the system clock to ensure that the data is latched while it is valid.

The 74F646 transceivers are enabled by the $\overline{\text{MEMOE}}$ signal. The direction of the device is selected by the memory write (MEMWR) signal. When the MEMWR signal is asserted, indicating that the CPU is writing to memory, the flow is directed from the AD lines to the MD lines (B-to-A). When the CPU reads from memory, the MEMWR signal is negated, and data flows in the opposite direction (A-to-B). The MEMWR signal is asserted one state before MEMOE is asserted and held valid for one state after MEMOE is negated. This is to avoid possible contention in the 74F646s which could result from switching the direction and enabling the device on the same clock edge.

The select pins (SAB and SBA) for the 74F646s determine for each direction whether data is latched in that direction or transferred through the device in real-time (i.e., the device is transparent). When SAB and/or SBA is negated, the device is transparent in the corresponding direction. When SAB or SBA is asserted, data is latched in the corresponding direction. For this design, the flow from the M bus to the DRAM array is latched; therefore, SBA is tied high. In the other direction, the device is transparent, so SAB is tied low.

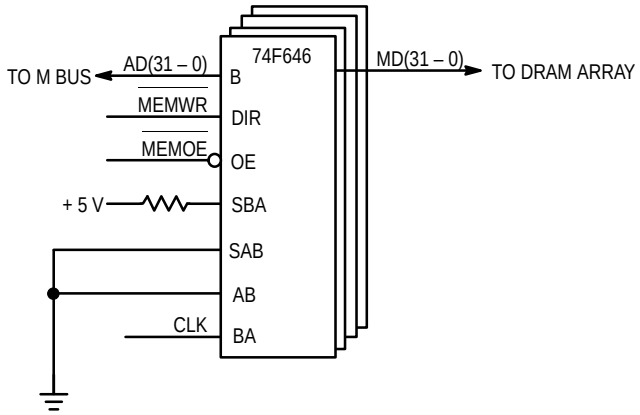

Figure 8. Data Latching

PARITY

The MC88200 CMMU devices have an automatic parity generation/checking feature which, on writes, generates parity for both addresses and data, and on reads, checks parity for both addresses and data. This feature can only be enabled or disabled for both addresses and data, not for only addresses or only data.

Since this DRAM interface system was originally part of a larger application, it was designed so that there could be another bus master in addition to the two MC88200s. To maintain parity detection, the other M bus master needed a method of generating a parity bit for each byte it wrote to memory, so parity generation/checking circuitry was added to the interface system for use by all of the possible bus masters. Therefore, in the design presented in this document, the automatic parity checking of the MC88200s has been disabled, and the MC88200s use the added parity circuitry. MC88200 automatic parity checking is disabled by clearing the parity enable bit (bit 15) in the MC88200 system control register (SCR).

Parity evaluation for an MC88200 system can be performed by hardware or software. Software parity detection requires no extra board space, but does not keep an accurate record of parity errors as hardware detection can. With software parity detection, when the processor receives the bus error signal, software diagnostic code is executed. This code may simply reissue the instruction; however, no record of the error is kept. If the parity error does not occur on the next access, this method is successful, but if the instruction causes a parity error again, it may be necessary to read each byte of data separately to determine which byte generated the error.

Hardware parity evaluation requires more board space than software parity evaluation but keeps an accurate record of the occurrence of parity errors. When performing hardware detection, an error can be recorded as soon as it occurs. This ability provides an advantage over software parity detection since software detection may not be able to reproduce an intermittent error for diagnostic purposes. Because of this advantage, hardware parity detection was chosen for this interface system.

Figure 9 shows the parity generation/checking circuitry implemented in this system. Parity evaluation is performed on each byte of data by four 9-bit parity generator/checkers (74AS280s) using odd parity. On write operations, all data going to the DRAMs (on MD31 – MD0) from the 74F646s is fed to the 74AS280s which perform parallel parity generation for the four bytes of data. On read operations, all data read from the DRAMs is checked for parity errors in parallel by the AS280s. The MD31 – MD0 lines are connected to 4.7 kΩ pull-up resistors to reduce the noise caused by rapid switching of the AS280s. The AS280s switch rapidly anytime the MD31 – MD0 lines three-state.

The 74AS280s check for parity during the data phase of a cycle and generate four parity error signals (PEA – PED) which indicate whether or not there was a parity error for each SIMM (A – D) in a bank. PEA – PED are asynchronous signals and valid for only one clock period. Since it is possible, in the worst case, that these signals will not meet the setup time for a PAL, they are first latched by a 74F374. The 74F374 is used because it has a setup time of only 2 ns. The 74F374 output enable pin is connected to ground (i.e., outputs are enabled), and the clock is connected to the system clock. Therefore, the F374 latches the parity error data and holds it for one clock period.

The outputs of the 74F374s ($\overline{PA}$ – $\overline{PD}$) are sent into the PARITY PAL where they are checked for valid parity. The PARITY PAL only checks PA – PD while the CHECK signal is valid, as shown in Figure 10. The CHECK signal is asserted at the end of the CAS signal, after valid data has been checked by the AS280s for parity errors.

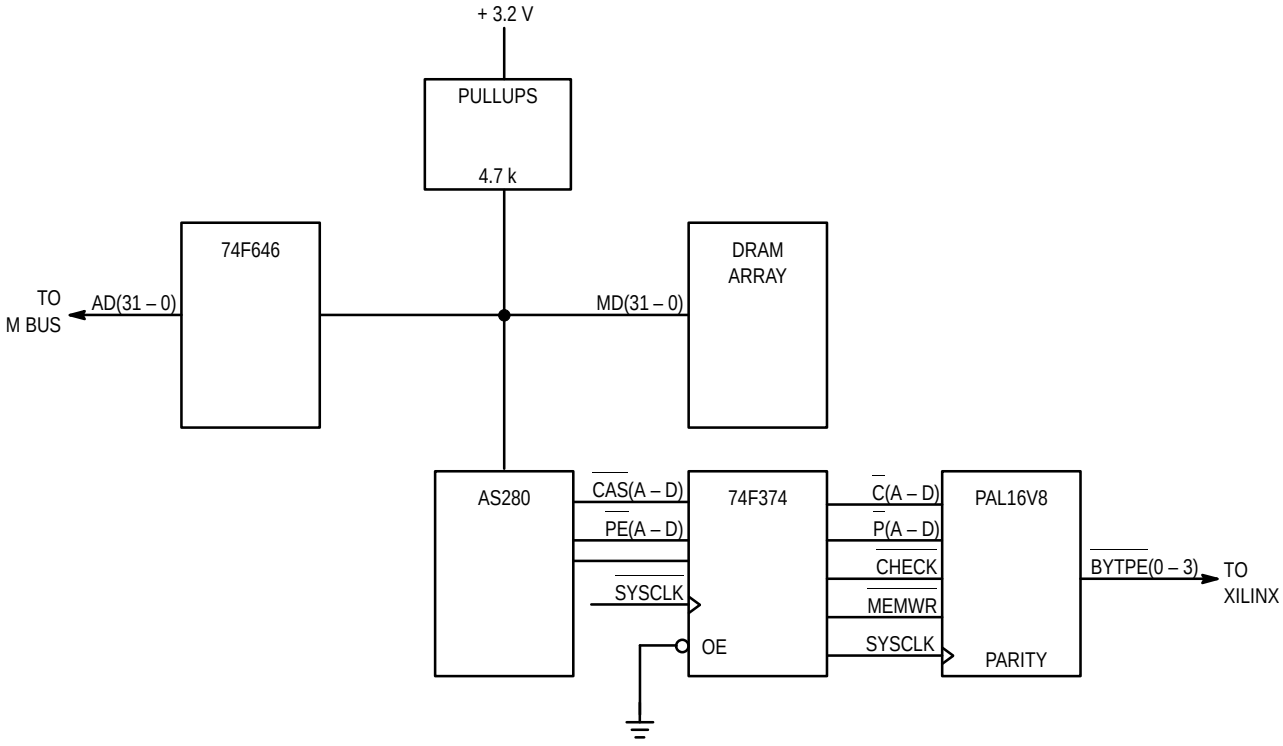

Figure 9. Parity Generation/Checking Circuitry

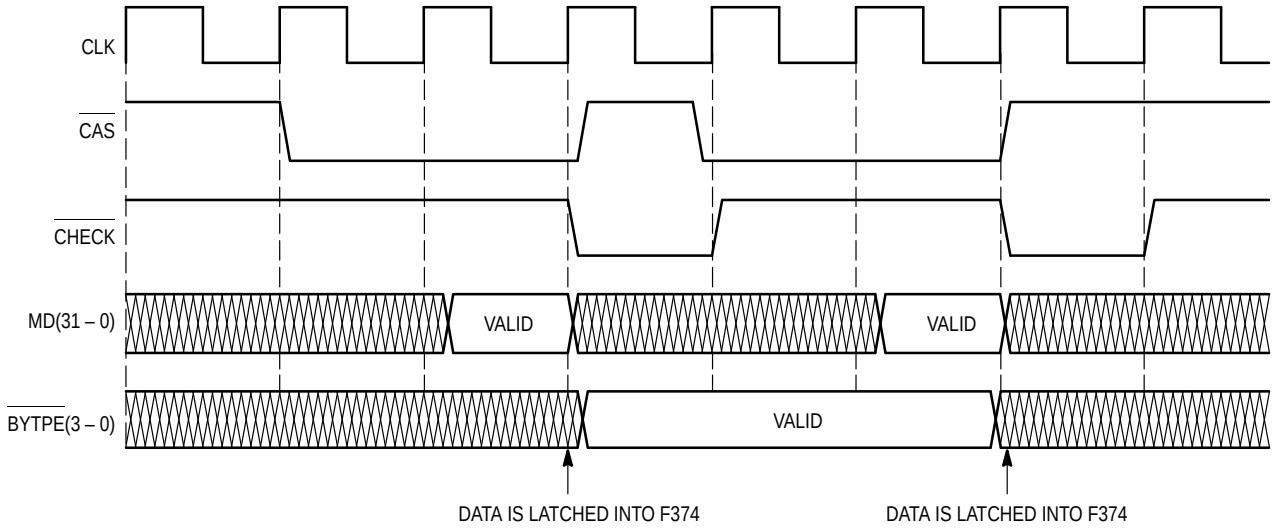

Figure 10. Parity Checking Timing

If a parity error is indicated during the time the CHECK signal is active, the PARITY PAL asserts the appropriate BYTPE0 – BYTPE3 signal(s) to indicate which of the SIMMs produced the parity error. The BYTPE0 – BYTPE3 signals are connected to a Xilinx gate array which controls parity errors in addition to other functions which are not related to this memory interface. The BYTPE0 – BYTPE3 signals are held valid for three clock periods to allow the Xilinx to latch the information.

The memory parity status register (MPSR) in the Xilinx (see Figure 11), keeps track of all parity information. Four bits in the MPSR keep track of which bytes, if any, generated a parity error, and two bits keep track of which bank contained the SIMM with the error.

When a memory parity error is detected, the appropriate bits in the MPSR are set, and the Xilinx sends a parity interrupt signal to a central interrupt controller. This interrupt controller is a peripheral chip which is external to the memory interface. When the interrupt controller receives a parity interrupt from the Xilinx, it sends an interrupt to the CPU which forces the CPU to execute an interrupt service routine. The interrupt controller has registers which contain information about the interrupt. The CPU reads the status of these registers to determine which device caused the interrupt: in this case, the Xilinx. Next, the CPU reads the MPSR in the Xilinx to determine which bank of memory and which byte(s) had the parity error thus indicating which SIMM device generated the error. Reading the MPSR clears the register and clears the interrupt in the Xilinx.

Once the CPU has determined that the interrupt was caused by a parity error, exception processing software should take over. This software can log the occurrence of parity errors for a system administrator or notify the user that a parity error occurred. This software can also reissue the access which caused the error or run diagnostic tests on the memory system.

REFRESH CYCLE

The 1-Mbit and 4-Mbit DRAM refresh specification is 512 or more refresh cycles per 8 ms. To meet this specification, one refresh request must be sent at least every 15.625 µs. The time between refresh requests must allow for the longest amount of time any bus master can have the bus since refresh requests cannot interrupt a bus master's tenure. The refresh request (REF) signal for this design is generated every 15 ms by a peripheral device which is not a part of the memory interface.

To reduce the amount of hardware required to support the refresh operation, the control logic generates a CAS-before-RAS refresh. Externally generated addresses are not necessary for a CAS-before-RAS refresh (as they are in a RAS-only refresh) because any external address is ignored and the refresh address is generated internally.

The peripheral device that generates the REF signal is clocked by a clock with a different frequency than the system clock. To insure that the REF signal does not become metastable, the REF signal is sent through a 74F374 D flip-flop (see Figure 12) twice. The synchronized REF signal is then input to the MODE PAL which generates the MODE1 and MODE0 signals as outputs. The MODE1 and MODE0 signals are inputs to the CAS PAL and indicate whether a read/write access or a refresh is to be performed (see Table 4).

7	6	5	4	3	2	1	0
NA	NA	BANK1	BANK0	BYTE3	BYTE2	BYTE1	BYTE0

Figure 11. Memory Parity Status Register

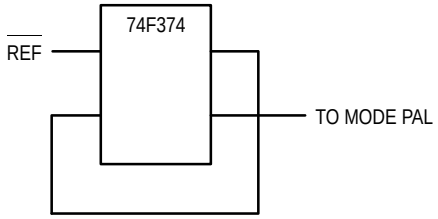

Figure 12. $\overline{\text{REF}}$ Synchronization by a D Flip-Flop

Table 5. $\overline{\text{MODE1}}$ and $\overline{\text{MODE0}}$ Function Description

MODE0	MODE1	Description
0	0	Refresh
0	1	NA
1	0	Read/Write
1	1	Idle

When both $\overline{\text{MODE}}$ signals are low, the CAS PAL asserts CAS to indicate that a refresh operation is to be performed. Next, the RAS PAL asserts the RAS0 signal to refresh bank0, and then one clock later the RAS PAL asserts RAS1 to refresh bank1 (see Figure 13). In this manner, both banks of DRAMs go through a refresh cycle. The two RAS signals are staggered by one clock so that both banks of DRAMs are not activated at the same time. Activating both banks at the same time causes a large current draw; therefore staggering the refresh operation minimizes current draw, which allows flexibility in choosing a power supply.

DRAM CONTROL LOGIC

The DRAM control logic consists of two parts (the state machine and the main memory control) which are implemented with 15 ns PAL devices. The overall functionality of each PAL is discussed in the following paragraphs. For more detailed information about the individual signals and states generated by the PALs, refer to **LIST OF LOGIC EQUATIONS** for the PAL logic equations.

The state machine is contained in the STATE PAL. The state machine generates 32 states which are output on pins Q0 – Q4. The 32 states are generated based on C0, C1, WRITE, and the previous state of the and MODE1 and MODE0 signals (see Figure 14). The MODE PAL generates the MODE1 and MODE0 signals based on the current state.

The WRITE PAL asserts the clocked WRITE signal when it receives either a data bus acknowledge or an instruction bus acknowledge signal and control signal C2 is negated (indicating a write).

The state machine changes state on the rising edge of every clock except when it is in state 0 (Idle) and WRITE and C0 are negated. Since the state machine changes states on every clock, the amount of external hardware needed is minimized. For example, during a burst cycle, no external counter is needed to keep track of the number of clocks that have passed. Instead the state machine replaces the counter since it steps through the states which indicate exactly which part of the transaction is being performed.

Nineteen of the 32 states are used during burst accesses, and seven other states are entered during refresh operations. The remaining six states are not used for the DRAM interface and are not presented here. The OK and WAIT signals are generated for the memory sub-system by the STATE PAL based on the status of the MODE1 and MODE0 signals and on control signals C0 and C1.

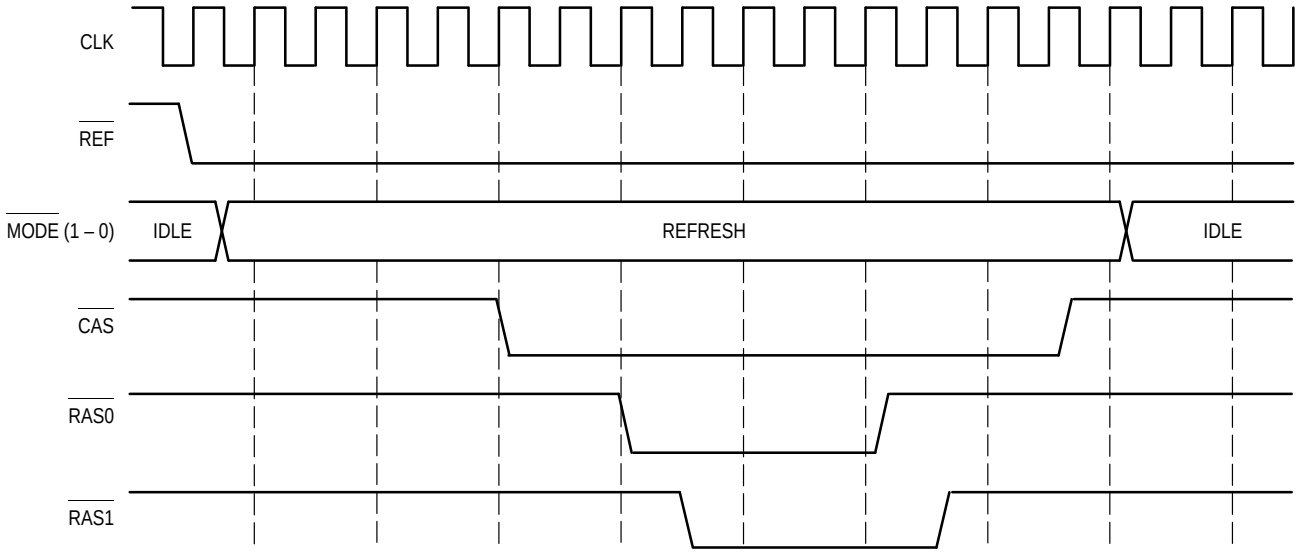

Figure 13. Refresh Cycle Timing Diagram

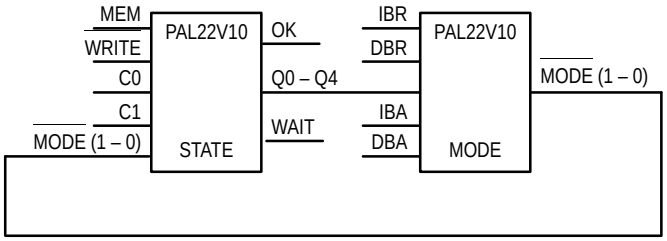

Figure 14. State Machine

Because the state machine is so large, its state diagram cannot be shown in detail. However, Figure 15 shows the flow of the states for read and write operations, including bursting.

The main memory control is made up of the RAS/CAS PAL, the RAS PAL, and the CAS PAL (see Figure 16). The ENCAS signal is generated by the RAS/CAS PAL based upon the current state. The ENCAS signal along with one or more of control signals C3 – C6 causes the CAS PAL to assert the appropriate CASA – CASD signal(s). The ENCAS signal remains asserted for two clock periods for all access types except refresh requests. Recall that the MODE1 and MODE0 signals indicate to the CAS PAL the access type.

The RAS1 and RAS0 signals are generated by the RAS PAL based upon the ENRAS, RAEND, BANK0, and BANK1 signals. The BANK0 and BANK1 signals are generated in the BANK PAL based upon the current address. Note that although only one BANK signal is actually necessary for this system (i.e. BANK=0 for bank0 and BANK=1 for bank1), two BANK signals were used to allow the flexibility to expand to four banks of DRAMs. The ENRAS signal along with BANK0 or BANK1 causes the RAS PAL to assert either RAS0 or RAS1, respectively. Refresh signaling makes generating RAS0 or RAS1 more complex than generating the CASA – CASD signals; therefore, the RAEND signal is used to minimize the number of OR terms in the PAL equations. RAEND negates RAS0 and RAS1.

When RAS0 or RAS1 is asserted, the ANYRAS signal is also asserted. When any of the CASA – CASD signals is asserted, the ANYCAS signal is also asserted.

Figure 17 shows the timing of the RAS and CAS signals for a burst read operation. A summary of the control logic signals is given in Table 6.

CONCLUSI0N

The memory system presented in this application note provides a high-performance, low cost design using either page- or nibble-mode DRAM technology. The system is a 4:3:3:3 for burst read accesses and a 5:3:3:3 for burst write accesses, adding latency to the minimum 2:1:1:1 access for an MC88100/88200 system. The system can use 1- or 4-Mbit DRAMs, allowing memory configurations from 4- to 32-Mbits. This system can run at 25 or 33.3 MHz, depending on the DRAM speed used. A higher frequency design could be implemented with interleaved banks of memory or a pipelined data path.

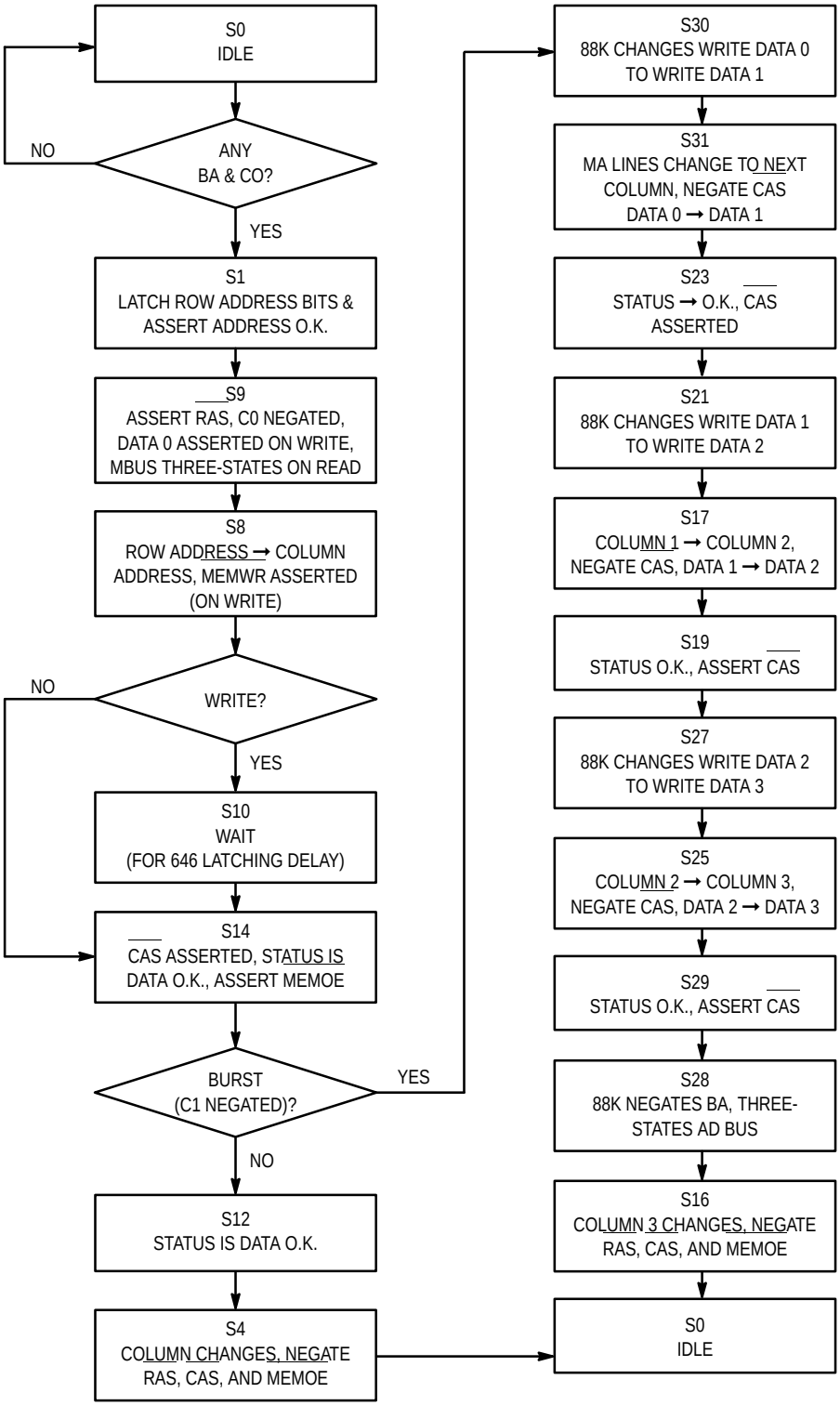

Figure 15. State Machine Flow Diagram

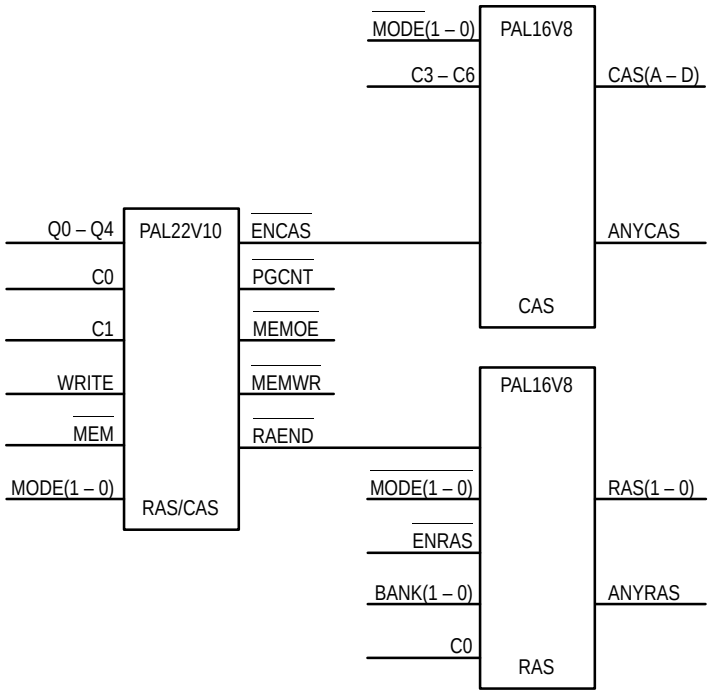

Figure 16. Main Memory Control

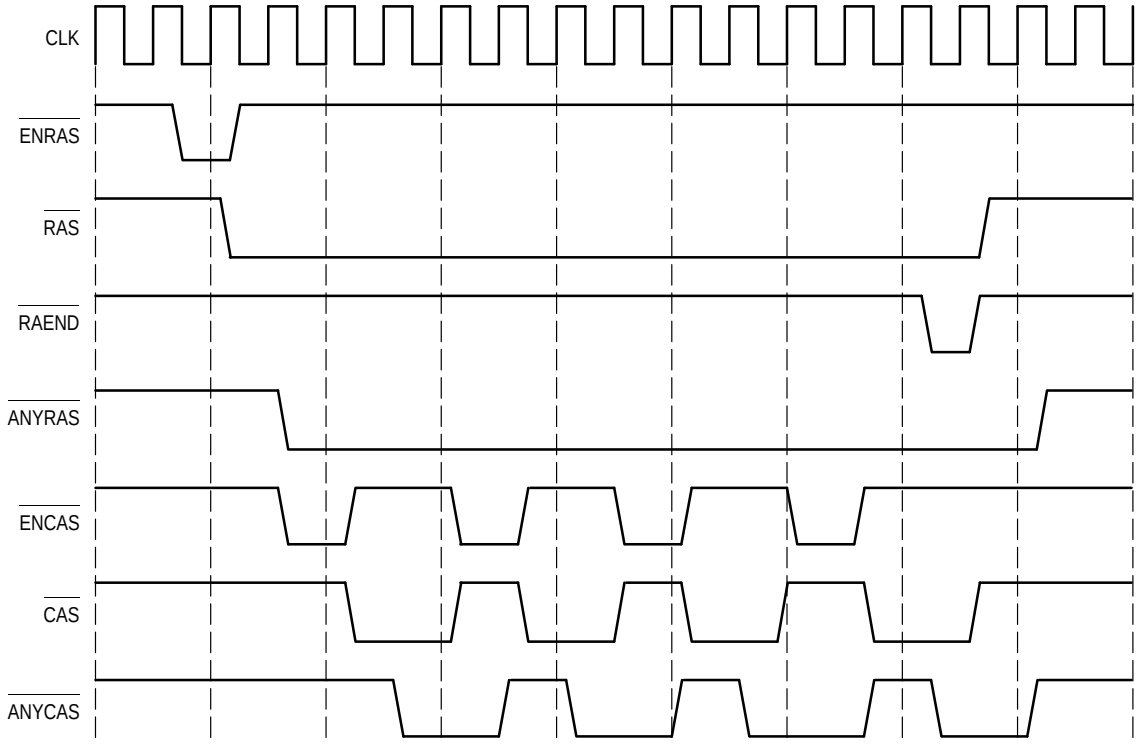

Figure 17. $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ Operation During a Burst Read

Table 6. Control Signal Summary

Signal	Description	Generating PAL
Q4 – Q0	The 32 states are output on these four signals. The state changes on every clock edge except when in state 0 with C0 and WRITE negated.	STATE
WRITE	Indicates a read/write access. Asserted when either data bus acknowledge or instruction bus acknowledge is asserted and C2 is negated.	WRITE
OK	Asserted or negated depending on the MODE signals and C0 and C1. Indicates the status of the memory system during read/write accesses.	STATE
WAIT	Asserted or negated depending on the MODE signals and C0 and C1. Indicates the status of the memory system during read/write accesses.	STATE
MODE1 – MODE0	Asserted or negated depending on the state. These signals indicate the type of access (see Table 4).	MODE
MEMOE (Memory Output Enable)	Output enable control for the 74F646s that interface the DRAMs and M Bus. Asserted one state before data is valid and negated on the state after the data bus is three-stated.	RAS/CAS
MEMWR (Memory Write)	Controls the direction of data between the DRAM array and the M bus. This signal is buffered in the 74F244s along with MA10MA0. The buffered outputs from the PAL which correspond to MEMWR are XWE and YWE for the bank0 and bank1 R/W inputs, respectively. The MEMWR signal is asserted one state before MEMOE and held for one clock after MEMOE is negated.	RAS/CAS
PGCNT (Page Count)	Asserted during burst cycles when using page-mode DRAMs. This signal increments the column address to access the next byte of data.	RAS/CAS
ENCAS (Enable CAS)	Asserted or negated depending on the state. The ENCAS signal along with one or more of control signals C3 – C6 causes the CAS PAL to assert the appropriate CASA – CASD signal(s).	RAS/CAS
ENRAS (Enable RAS)	Asserted or negated depending on the state. The ENRAS signal along with BANK0 or BANK1 causes the RAS PAL to assert either RAS0 or RAS1, respectively.	88RAS
RAEND (End RAS)	Asserted or negated depending on the state. Causes the negation of RAS0 and RAS1.	RAS/CAS
ANYCAS	Asserted when any of the CASA – CASD signals is asserted.	CAS
ANYRAS	Asserted when RAS0 or RAS1 is asserted.	RAS

LIST OF LOGIC EQUATIONS

```
                NAME    BANK
                COMPANY         MOTOROLA
                PART            PAL16R6; 16V8
/*****************************************************************************/
/*      This device decodes the bank outputs. BANK0 and BANK1 control      */
/*      which RAS signal is asserted. 1 MBIT indicates whether the DRAM is */
/*      1 or 4 Mbit. C0 determines when the address is valid. This PAL also */
/*      latches AD19, AD20, and AD21                                       */
/*****************************************************************************/
/**     Inputs          **/
CLK             /* System clock                                           */
C0              ; M Bus control signal that determines when address is valid
1MBIT           ; Input from jumper. Indicates 1 Mbit or 4 Mbit DRAMs.  */
/AD19           ; Muxed address/data line 19                               */
/AD20           ; Muxed address/data line 20                               */
/AD21           ; Muxed address/data line 21                               */
/AD22           ; Muxed address/data line 22                               */
/AD23           ; Muxed address/data line 23                               */
/AD24           ; Muxed address/data line 24                               */
/AD25           ; Muxed address/data line 25                               */
/**     Outputs         **/
/BANK0                  /* Bank 0 selected                                          */
/BANK1                  /* Bank 1 selected                                          */
/LA19           /* Latched address 19                                     */
/LA20           /* Latched address 20                                     */
/LA21           /* Latched address 21                                     */
/**     Logic Equations         **/
BANK0           =       1MBIT & C0 & /AD22
                        + /1 MBIT & C0 & /AD24
                        + BANK0 & /C0
BANK1           =       1MBIT & C0 & /AD23
                        + /1 MBIT & C0 & /AD25
                        + BANK1 & /C0
LA19            =       C0 & AD19
                        + /C0 & LA19
LA20            =       C0 & AD20
                        + /C0 & LA20
LA21            =       C0 & AD21
```

```
                NAME    PARITY
                COMPANY         MOTOROLA
                PART            PAL16V8
/*****************************************************************************/
/*      This device generates the parity error signals for each byte during a */
/*      memory read operation. Errors are asserted for 2 clock periods.    */
/*      Error information is latched in the Xilinx. ANYPE is generated to aid in */
 /*     troubleshooting.                                                   */
/*****************************************************************************/
/**     Inputs          **/
CLK             /* System clock                                           */
```

```
/CA             /* Cas signal for byte A                                    */
/CB             /* Cas signal for byte B                                    */
/CC             /* Cas signal for byte C                                    */
/CD             /* Cas signal for byte D                                    */
PA              /* Parity error signal for byte A                           */
PB              /* Parity error signal for byte B                           */
PC              /* Parity error signal for byte C                           */
PD              /* Parity error signal for byte D                           */
/CHK            : Check signal indicating when to check for errors */
/MEMWR          /* Indicates a write to memory                              */
/**      Outputs          **/
/BYTPE0         /* Indicates parity error on byte 0                         */
/BYTPE1
/BYTPE2
/BTYPE3
/ANYPE          /* Indicates a parity error on any byte                     */
/ANYPE1         /* Generated to hold signals an extra clock                 */
/**      Logic Equations          **/
BYTPE0          =       CHK & /MEMWR & CA & PA
                        + BYTPE0 & ANYPE
                        + BYTPE0 & ANYPE1
BYTPE1          =       CHK & /MEMWR & CB & PB
                        + BYTPE1 & ANYPE
                        + BYTPE1 & ANYPE1
BYTPE2          =       CHK & /MEMWR & CC & PC
                        + BYTPE2 & ANYPE
                        + BYTPE2 & ANYPE1
BYTPE3          =       CHK & /MEMWR & CD & PD
                        + BYTPE3 & ANYPE
                        + BYTPE3 & ANYPE1
ANYPE           =       CHK & /MEMWR & CA & PA
                        + CHK & /MEMWR & CB & PB
                        + CHK & /MEMWR & CC & PC
                        + CHK & /MEMWR & CD & PD
                        + ANYPE & /ANYPE1
ANYPE1          =       ANYPE                           + /C0 & LA21
```

```
                NAME    CAS
                COMPANY         MOTOROLA
                PART            PAL16R8
/*************************************************************************************/
/*      This device controls the registered CAS outputs.                             */
/*************************************************************************************/
/**      Inputs          **/
CLK             /* System clock                                                     */
/MODE0          /* Together, MODE0 and MODE1 indicate what                          */
/MODE1          /*      type of access                                              */
C3              /* M Bus byte enable 3                                              */
C4              /* M Bus byte enable 4                                              */
C5              /* M Bus byte enable 5                                              */
C6              /* M Bus byte enable 6                                              */
/ENCAS          /* Enable CAS signal                                                */
```

```
/**      Outputs          **/
/CASA          /* Registered CAS signal for bits 31 – 24                */
/CASB          /* Registered CAS signal for bits 23 – 16                */
/CASC          /* Registered CAS signal for bits 15 – 8                 */
/CASD          /* Registered CAS signal for bits 7 – 0                  */
/ANYCAS              /* Any CAS signal is asserted                          */
/**      Logic Equations          **/
CASA   =       ENCAS & MODE1 & /MODE0 & C3
                     + ENCAS & MODE 1 & MODE0
CASB    =      ENCAS & MODE1 & /MODE0 & C4
                     + ENCAS & MODE 1 & MODE0
CASC   =       ENCAS & MODE1 & /MODE0 & C5
                     + ENCAS & MODE 1 & MODE0
CASD   =       ENCAS & MODE1 & /MODE0 & C6
                     + ENCAS & MODE 1 & MODE0
ANYCAS         =       ENCAS & /MODE1 & MODE0
                     + ENCAS & MODE1 & /MODE0
                     + ENCAS & MODE1 & MODE0
```

```
                NAME   RAS
                COMPANY        MOTOROLA
                PART           PAL16R8
/*****************************************************************************/
/*      This device controls the registered RAS outputs. BANK1 and BANK0      */
/*      control which RAS is to be asserted. Each RAS is staggered during     */
/*      refresh by one clock. This PAL also latches AD3 – AD5.                */
/*****************************************************************************/
/**      Inputs           **/
CLK            /* System clock                                          */
/MODE0          /* Together, MODE0 and MODE1 indicate what               */
/MODE1          /*      type of access                                   */
BANK0          /* Bank 0 selected                                       */
BANK1          /* Bank 1 selected                                       */
C0             /* M Bus control signal for address/data phase           */
/ENRAS         /* Enable RAS signal                                     */
/RAEND         /* RAS end signal                                        */
AD3            /* Multiplexed address/data line 3                       */
AD4            /* Multiplexed address/data line 4                       */
AD5            /* Multiplexed address/data line 5                       */
/**      Outputs          **/
/RAS0          /* Registered RAS signal for bank 0              */
/RAS1          /* Registered RAS signal for bank 1              */
/ANYRAS        /* Any RAS signal is asserted                        */
/LA3            /* Latched address line 3                                */
/LA4            /* Latched address line 4                                */
/LA5            /* Latched address line 5                                */
/**      Logic Equations          **/
RAS0   =       ENRAS & MODE1 & MODE0
                     + ENRAS & /MODE 1 & MODE0 & /BANK1 & /BANK0
                     + ENRAS & MODE1 & /MODE0 & /BANK1 & /BANK0
                     + RAS0 & /RAEND
```

```
RAS1    =       RAS0 & MODE1 & MODE0
                + ENRAS & /MODE1 & MODE0 & /BANK1 & BANK0
                + ENRAS & MODE1 & /MODE0 & /BANK1 & BANK0
                + RAS1 & /MODE1 & MODE0 & /RAEND
                + RAS1 & MODE1 & /MODE0 & /RAEND
ANYRAS          =       ENRAS
                + RAS0 & MODE1 & MODE0
                + RAS1 & MODE1 & MODE0
                + ANYRAS & /MODE1 & MODE0 & /RAEND
                + ANYRAS & MODE1 & /MODE0 & /RAEND
LA3     =               C0 & AD3
                + /C0 & LA3
LA4     =               C0 & AD4
                + /C0 & LA4
LA5     =               C0 & AD5
                + /C0 & LA5
```

```
                NAME    ARB
                COMPANY         MOTOROLA
                PART            PAL16L8 ; 16V8
/*************************************************************************/
/*      This device arbitrates between the data CMMU and instruction CMMU.   */
/*      Priorities are as follows:                                            */
/*      1 Refresh               (RREQ)                          */
/*      2 Instruction CMMU   (IBG)                              */
/*      3. Data CMMU         (DBG)                              */
/*************************************************************************/
/**     Inputs          **/
CLK             /* System clock                                          */
/MODE0           /* Together, MODE0 and MODE1 indicate what   */
/MODE1           /*      type of access                               */
IBR             /* Instruction bus request from the instruction CMMU     */
DBR             /* Data bus request from the data CMMU                   */
/RREQ            /* Refresh request                                       */
/**     Outputs         **/
/DBG             /* Data bus grant                                        */
/IBG             /* Instruction bus grant                                 */
/**     Logic Equations         **/
DBG     =       MODE0
                + MODE1
                + RREQ
                + IBR
                + /DBR
IBG     =       MODE0
                + MODE1
                + RREQ
                +/IBR
```

```
                NAME    RAS/CAS
                COMPANY         MOTOROLA
                PART            PAL22V10
/*************************************************************************/
/*      This device controls the main memory interface.                  */
/*************************************************************************/
/**     Inputs          **/
```

```
CLK             /* System clock                                          */
/MODE0          /* Together, MODE0 and MODE1 indicate what   */
/MODE1          /*      type of access                                   */
C0              /* M Bus control signal indicating read or write cycle   */
C1      /* MBus control signal indicating the end of a burst transfer  */
/WRITE          /* Indicates a read/write access                         */
/MEM            /* Address is in memory space                            */
Q0              /* Q0 – Q4 indicate which state the state machine is in  */
Q1
Q2
Q3
Q4
/**     Outputs         **/
/RAEND          /* End RAS signal                                        */
/ENCAS          /* Enable CAS signal                                     */
/MEMWR          /* Indicates a write to memory                           */
/MEMOE  /* Output enable for 646 latches between MBus and DRAMs  */
/PGCNT          /* Used to toggle address lines for page-mode DRAMs  */
/**     Logic Equations         **/
ENCAS   =       /Q1 & /Q2 & Q3 & /Q4 & MEM & /WRITE
                + /Q2 & Q3 & Q4 & MEM & Q0
                + /Q1 & /Q2 & Q3 & /Q4 & MODE1 & MODE0
                + /Q2 & Q3 & /Q4 & MEM & WRITE & /Q0
                + /Q1 & /Q4 & Q0 & MODE1 & MODE0
                + Q1 & Q2 & Q3 & /Q4 & MODE1 & MODE0
                + /Q1 & /Q3 & Q4 & MEM & Q0
                + /Q2 & /Q4 & Q0 & MODE1 & MODE0
                + Q1 & Q2 & Q3 & Q4 & MEM
MEMOE   =       WRITE & Q0 & Q3 & MEM & /MODE0
                + Q1 & Q2 & MEM & /MODE0
                + /Q0 & /Q2 & Q3 & MEM & /MODE0
                +Q0 & Q4 & MEM & /MODE0
MEMWR   =       MEMWR & MEMOE
                + WRITE & Q0 & /Q4 & MEM & /MODE0
PGCNT   =       /Q0 & Q1 & Q2 & Q3 & /Q4 & MODE1 & /MODE0 & /C1
                + Q0 & Q1 & /Q3 & Q4
RAEND   =       /Q0 & Q1 & Q2 & Q3 & /Q4 & MODE1 & /MODE0 & C1
                + /Q0 & Q1 & Q2 & Q3 & /Q4 & /MODE1 & MODE0
                + Q0 & Q1 & Q2 & Q3 & /Q4 & MODE1 & MODE0
                + Q0 & /Q1 & Q2 & Q3 & Q4
.
```

```
                NAME    MODE
                COMPANY         MOTOROLA
                PART            PAL22V10
/*****************************************************************************/
/'      This device determines which “master” is in control of the current cycle.  */
/*      It also generates the refresh request for the DRAM.                    */
/*****************************************************************************/
/**     Inputs          **/
CLK             /* System clock                                          */
/MODE0          /* Together, MODE0 and MODE1 indicate what   */
```

```
/MODE1          /*      type of access                                  */
Q0              /* Q0 – Q4 indicate which state the state machine is in  */
Q1
Q2
Q3
Q4
WRITE           /* Indicates a read/write access                        */
/MEM            /* Address is in memory space                           */
AD29            /* Multiplexed address/data line 29                     */
AD31            /* Multiplexed address/data line 31                     */
/**     Outputs         **/
/ENRAS          /* Enable RAS signal.                                   */
CHECK           /* Used for parity checking                             */
/**     Logic Equations         **/
ENRAS   =       /Q0 & /Q1 & /Q2 & /Q3 & /Q4 & MODE1 & /MODE0 & /AD31
                & AD20 & C0
                + Q0 & Q1 &/Q2 &    & /Q4 & .MODE1 & MODE0 & MEM
                + Q0 & Q1 & Q3 & /Q4 & MODE1 & MODE0
CHECK   =       /Q0 & /Q1 & Q2 & Q3 & /Q4 & MODE1 & /MODE0 & MEM
                + /Q0 & /Q1 & /Q2 & /Q3 & Q4 & MODE1 & /MODE0 & MEM
                + Q0 & /Q1 & /Q2 & /Q3 & Q4 & MODE1 & /MODE0 & MEM
                + Q0 & /Q1 & /Q2 & Q3 & Q4 & MODE1 & /MODE0 & MEM
                + Q0 & Q1 & Q2 & Q3 & Q4 & MODE1 & /MODE0 & MEM
```

```
                NAME    AD2MA
                COMPANY         MOTOROLA
                PART            PAL16R6; 16V8
/*****************************************************************************/
/'      This device multiplexes the row and column bits to form the outputs     */
/'      MA5 through MA8. During the address phase (C0 high),                    */
/'      the ADXX signals become the row address. When ANYRAS                    */
/'      is asserted, the LAXX signals become the column address. The column     */
/'      is held until the end of the memory cycle (ANYRAS is negated). This     */
/'      device also latches AD13 and AD14.                                      */
/*****************************************************************************/
/**     Inputs          **/
CLK             /* System clock                                         */
C0              /* M Bus control signal for address/data phase          */
AD9             /* Multiplexed address/data line 9                      */
AD10            /* Multiplexed address/data line 10                     */
AD11            /* Multiplexed address/data line 11                     */
AD12            /* Multiplexed address/data line 12                     */
AD13            /* Multiplexed address/data line 13                     */
AD14            /* Multiplexed address/data line 14                     */
LA20            /* Latched address line 20                              */
LA21            /* Latched address line 21                              */
/**     Outputs         **/
LA13            /* Latched address line 13                              */
LA14            /* Latched address line 14                              */
MA5             /* Multiplexed address line 5                           */
MA6             /* Multiplexed address line 6                           */
MA7             /* Multiplexed address line 7                           */
```

```
MA8             /* Multiplexed address line 8                          */
/**     Logic Equations         **/
LA13    =       C0 & AD13
                + /C0 & LA13
LA14    =       C0 & AD14
                + /C0 & LA14
MA5     =       C0 & /ANYRAS & AD9
                + ANYRAS & LA13
MA6     =       C0 & /ANYRAS & AD10
                + ANYRAS & LA14
MA7     =       C0 & /ANYRAS & AD11
                + ANYRAS & LA20
MA8     =       C0 & /ANYRAS & AD12
                + ANYRAS & LA21
```

```
                NAME    AD2MA/A
                COMPANY         MOTOROLA
                PART            PAL16R6; 16V8
/*****************************************************************************/
/'      This device multiplexes the row and column bits to form the outputs     */
/'      MA0, MA2, MA3, and MA10. During the address phase (C0 high),            */
/'      the ADXX signals become the row address. When ANYRAS                    */
/'      is asserted, the LAXX signals become the column address. The column     */
/'      is held until the end of the memory cycle (ANYRAS is negated).          */
/*****************************************************************************/
/**     Inputs          **/
CLK             /* System clock                                                */
C0              /* M Bus control signal for address/data phase         */
AD15            /* Multiplexed address/data line 15                    */
AD16            /* Multiplexed address/data line 16                    */
AD17            /* Multiplexed address/data line 17                    */
AD18            /* Multiplexed address/data line 18                    */
AD22            /* Multiplexed address/data line 22                    */
AD23            /* Multiplexed address/data line 23                    */
LA4             /* Latched address line 4                                      */
LA5             /* Latched address line 5                                      */
/**     Outputs         **/
MA0             /* Multiplexed address line 0                          */
MA2             /* Multiplexed address line 2                          */
MA3             /* Multiplexed address line 3                          */
MA10            /* Multiplexed address line 10                         */
LA18            /* Latched address line 18                             */
LA23            /* Latched address line 23                             */
/**     Logic Equations         **/
LA23    =       C0 & AD23
                + /C0 & LA23
LA18    =       C0 & AD18
                + /C0 & LA18
MA0     =       C0 & /ANYRAS & AD15
                + ANYRAS & LA22
MA2     =       C0 & /ANYRAS & AD16
                + ANYRAS & LA23
```

```
MA3         =     C0 & /ANYRAS & AD17
                  + ANYRAS & LA4
MA10        =     C0 & /ANYRAS & AD18
                  + ANYRAS & LA5
```

```
                  NAME    ADR2NIB
                  COMPANY         MOTOROLA
                  PART            PAL16R6; 16V8
/*****************************************************************************/
/'       This device latches LA2 and LA7 and controls page mode or nibble      */
/'       mode address operation. It also controls the multiplexing of the row  */
/'       and column memory address as well as the changing of the memory       */
/'       address during page mode.                                             */
/*****************************************************************************/
/**      Inputs            **/
CLK               /* System clock                                             */
C0                /* M Bus control signal for address/data phase        */
PGCNT             /* Signal used to increment column address                */
ANYRAS            /* Indicates that a RAS signal is asserted                */
PAGE              /* Signal from jumper indicating page or nibble mode DRAMs */
AD2               /* Multiplexed address/data line 2                        */
AD6               /* Multiplexed address/data line 6                        */
AD7               /* Multiplexed address/data line 7                        */
AD8               /* Muitiplexed address/data line 8                        */
LA3               /* Latched address line 3                                   */
LA19              /* Latched address line 19                                */
/**      Outputs           **/
LA2               /* Latched address line 2                                   */
LA7               /* Latched address line 7                                   */
MA1               /* Multiplexed address line 1                             */
MA4               /* Multiplexed address line 4                             */
MNIB              /* Multiplexed address line 9                             */
DRAS
/**      Logic Equations          **/
DRAS        =     ANYRAS
LA2         =     C0 & AD2
                  + /C0 & LA2
LA7         =     C0 & AD7
                  + /C0 & LA7
MA1         =     PAGE & C0 & /ANYRAS & AD7
                  + PAGE    & ANYRAS & /LA2 & /PGCNT & /DRAS
                  + PAGE & /C0 & ANYRAS & /MA1 & PGCNT & DRAS
                  + PAGE & /C0 & ANYRAS & MA1 & /PGCNT & DRAS
                  + /PAGE & C0 & /ANYRAS & AD6      /* nibble mode      */
                  + /PAGE & /C0 & ANYRAS & LA7
MA4         =     PAGE & C0 & /ANYRAS & AD6
                  + PAGE    & ANYRAS & LA3 & /PGCNT & /DRAS
                  + PAGE & /C0 & ANYRAS & MA4 & /PGCNT & DRAS
                  +PAGE & /C0 & ANYRAS & /MA4 & PGCNT & DRAS & MA1
                  + PAGE & /C0 & ANYRAS & MA4       & DRAS &/ MA1
                  + /PAGE & C0 & /ANYRAS & AD8      /* nibble mode      */
                  + /PAGE & /C0 & ANYRAS & LA19
```

```
MNIB      =     PAGE & C0 & /ANYRAS & AD8
                + PAGE   & ANYRAS & LA19
                + /PAGE C0 & /ANYRAS & AD2      /* nibble mode      */
                + /PAGE   & ANYRAS & LA3
```

```
                NAME    WRITE
                COMPANY         MOTOROLA
                PART            PAL22V10
/*****************************************************************************/
/'      This device decodes the physical memory address space and also      */
/'      generates the WRITE signal.                                         */
/*****************************************************************************/
/**     Inputs              **/
CLK             /* System clock                                            */
/MODE0          /* Together, MODE0 and MODE1 indicate what    */
/MODE1          /*      type of access                                     */
C0              /* M Bus control signal for address/data phase      */
C2              /* M Bus control signal indicating memory read/write   */
AD29            /* Multiplexed address/data line 29                        */
AD30            /*      "           "       "  30                         */
AD31            /*      "           "       "  31                         */
/SRST           /* System reset signal                                     */
/**     Outputs             **/
/MEM            /* Indicates address is a valid memory address             */
WRITE           /* Indicates memory write                                  */
/**     Logic Equations         **/
MEM     =       /SRST & /AD31 & /AD30 & AD29 & C0
                + /SRST & MEM & /MODE1 & MODE0
                + /SRST & MEM & MODE1 & /MODE0
WRITE   =       /SRST & C2 & C0
                + /SRST & WRITE & /MODE1 & MODE0
                + /SRST & WRITE & MODE1 & /MODE0
```

```
                NAME    STATE
                COMPANY         MOTOROLA
                PART            PAL22V10
/*****************************************************************************/
/'      This device holds the system state machine. It also controls the WAIT   */
/'      and OK signals to the 88K.                                                  */
/*****************************************************************************/
/**     Inputs              **/
CLK             /* System clock                                            */
C0              /* M Bus control signal indicating read or write cycle     */
C1              /* M Bus control signal indicating the end of a burst transfer  */
/MEM            /* Indicates a write to memory                             */
/WRITE          /* Indicates a read/write access                           */
/MODE0          /* Together, MODE0 and MODE1 indicate what                 */
/MODE1          /*      type of access                                     */
/**     Outputs             **/
Q0              /* Q0–Q4 indicate which state the state machine is in      */
Q1
Q2
```

```
Q3
Q4
OK              /* Used to tell whether the memory transaction was        */
                /* completed successfully                                 */
/**       Logic Equations          **/
Q0        =     Q0 & /Q1 & Q2 & /Q4 & MODE1 & MODE0
                + Q0 & /Q1 & /Q3
                + Q0 & /Q1 & Q2 & /Q4 & MODE1 & WRITE
                + Q0 & Q1 & Q4
                + /Q0 & /Q1 & /Q2 & /Q3 & /Q4 & MODE1 & C0
                + Q1 & Q2 & Q3 & MODE1 & MODE0
                +/Q1 & /Q2 & /Q3 & /Q4 & MODE1 & MODE0
                + Q1 & Q2 & Q3 & Q4
                + Q0 & Q1 & Q3
                + Q0 & /Q2 & Q4
                + Q0 & Q1 & /Q2
                + Q0 & /Q1 & Q2 & /Q3
Q1        =     MODE1 & MODE0 & Q0 & /Q2 & /Q3
                + MODE1 & MODE0 & /Q0 & Q1 & Q2 & Q3
                + MODE1 & /MODE0 & /Q0 & Q1 & /Q2 & /Q4
                + MODE0 & Q0 & /Q2 & /Q3 & MEM
                + Q0 & /Q2 & /Q3
                + Q0 & Q1 & /Q2 & /Q3
                + /Q0 & /Q2 & Q3 & /Q4
                + Q1 & Q2 & Q3 & Q4
                + Q0 & /Q2 & /Q3 & Q4
                + Q0 & Q2 & /Q3 & /Q4
Q4        =     /Q0 & Q3 & /Q4 & MODE0 & /MEM
                + Q2 & Q3 & /Q4
                + /Q0 & Q3 & /Q4 & MODE0 & MODE1
                + /Q0 & Q3 & /Q4 & /MODE0 & /MODE1
                + Q0 & Q1 & Q2
                + Q0 & /Q1 & Q3 & Q4
                + /Q0 & Q3 & /Q4 & /WRITE
                + Q0 & Q2 & /Q4
                + Q1 & Q2 & /Q4
                + Q1 & Q2 & Q3
                + /Q0 & Q1 & Q3 & /Q4
Q3        =     Q0 & /Q1 & Q3 & /MODE0 & /WRITE
                + Q0 & /Q1 & Q3 & /MODE1
                + /Q2 & Q3 & /Q4
                + Q0 & /Q2 & /Q4 & /MODE1 & /MEM
                + Q0 & /Q2 & /Q4 & /MODE0 & /MODE1
                + /Q0 & Q1 & Q2 & Q3
                + Q0 & Q1 & /Q2
                + Q0 & /Q1 & Q3 & Q4
                + Q1 & Q3 & /Q4
Q4        =     /Q0 & Q1 & Q2 & Q3 & /C1 & /MODE0
                + Q0 & Q4
                + Q2 & Q3 & Q4
OK        =     MODE1 & /MODE0 & MEM & /Q0 & Q1 & Q2 & Q3 & /Q4 & C1
                + MODE1 & /MODE0 & MEM & Q0 & /Q1 & /Q2 & Q3 & Q4 & WRITE
                + MODE1 & /MODE0 & MEM & Q0 & /Q1 & Q2 & Q3 & Q4 & /WRITE
```

```
WAIT    =   /MODE1 & MODE0 & Q0 & /Q1 & /Q2 & /Q3 &1Q4 & MEM
            + MODE1 & /MODE0 & Q0 & Q1 & Q2 & Q3 & Q4 & MEM & WRITE
            + MODE1 & /MODE0 & /Q0 & Q1 & /Q2 & Q3 & /Q4 & MEM & WRITE
            + MODE1 & /MODE0 & /Q0 & /Q1 & /Q2 & /Q3 & /Q4 & C0
            + MODE1 & /MODE0 & /Q0 & Q1 & Q2 & Q3 & /Q4 & MEM & /WRITE
            + MODE1 & /MODE0 & Q0 & /Q1 & Q2 & Q3 & Q4 & MEM & /WRITE
            + MODE1 & MODE0 & Q0 & Q1 & /Q3 & Q4 & MEM & /WRITE
            + MODE1 & /MODE0 & Q0 & /Q1 & /Q2 & Q4 & MEM & WRITE
            + MODE1 & /MODE0
```

Literature Distribution Centers:
USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.
EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.
JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141, Japan.
ASIA PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.