Order this document
by AN1807/D

AN1807

Using Motorola's Dual Port NetRAMs for Interprocessor Communication in a Datacomm Application

Prepared by: Mike Parks, Motorola SPS, Austin, TX

INTRODUCTION

Dual port static RAMs are commonly used for communications between computing elements in a multiprocessor system. This application note describes the use of Motorola's dual port NetRAM products to provide a bidirectional pathway between an MPU controlling a datacomm switch and an ASIC that performs transformations on the data packets passing through the switch. In this particular example, the MPU is an MPC755 that uses its backside level–2 (L2) cache port to access the NetRAM. A single clock NetRAM, such as the MCM63D736, can be used if the ASIC clock can be derived from the L2 port clock (see AN1779/D for details). If independent clocks drive the two computing elements, the NetRAM must be one of the dual clock parts, such as the MCM63V736.

The described system uses circular buffers to ensure the transfer of packet data is performed correctly. The dual clock NetRAMs allow simultaneous accesses to the same address from each port. When the rising edges of the two clocks are nearly coincident, either access may occur first. As a result, the system must provide a higher order mechanism to ensure the proper data is used by the MPU and ASIC. In this example, the circular buffers provide the signaling between the two computing elements to synchronize the transfer of data in each direction. The chosen algorithm supports a single priority of data traffic, although a multiple priority extension is also briefly described.

SYSTEM DESCRIPTION

Figure 1 shows a simple block diagram of the relevant portion of the system architecture. Two 128K x 36 dual clock NetRAMs provide a 64–bit communications path between an MPC755 microprocessor and an ASIC or DSP. The MPU transfers 64–byte (8–word) data packets through the NetRAMs for processing by the ASIC. The transformed packets are then sent back through the NetRAMs to the MPU. The connection between the MPC755 and the NetRAMs uses the backside cache port of the processor. In the MPC755, the cache controller can be turned off so the port can be used as a high speed, point–to–point link. The bus interface is also programmable, allowing it to be matched to the pipelined interface of the NetRAM.

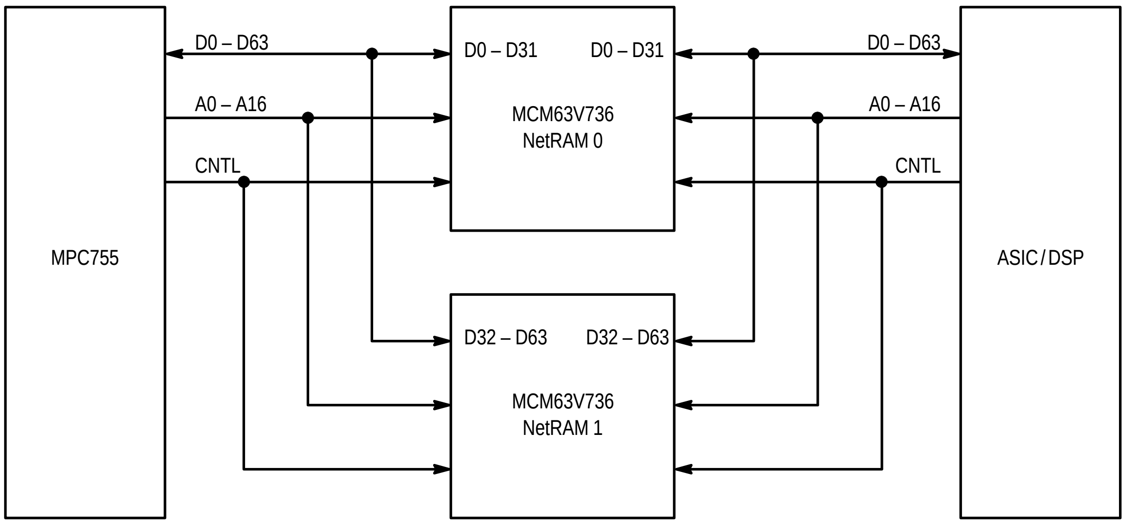

Figure 1. System Block Diagram

8/4/99

CIRCULAR BUFFER DESCRIPTION

Both the MPU and the ASIC maintain a circular buffer that facilitates the handshaking for the flow of packets to the other computing element. Each buffer consists of 8K words of 32 bits, residing in NetRAM 1 (see Figure 2 for the system's memory map). After the MPU transfers a 64–byte cell to the packet data memory, it places the address of the cell in the next available location of its circular buffer. The most significant bit of each circular buffer entry is a valid bit. A 1 indicates that a packet is available for the other computing element to retrieve. A 0 indicates that the packet has been read by the receiving entity.

The computing element that initiates the transfer of a packet (e.g., the MPU in the case of MPU–to–ASIC traffic), maintains a head pointer and a tail pointer to the circular buffer. The head pointer indicates the address of the oldest unprocessed packet in memory, while the tail pointer indicates the next entry that will be used to hold the address of a newly transferred packet. When the head or tail pointer reaches the end of the section of memory reserved for the circular buffer, it will wrap around to the beginning of the buffer when the next advance occurs.

The receiver of the packets maintains a single pointer to the next entry in the buffer that will need its attention. If the receiving entity has processed all the available packets, this read pointer is a copy of the tail pointer. Otherwise, it lags the tail pointer by a number of entries equal to the number of unprocessed packets.

Figure 3 illustrates the use of the MPU–to–ASIC circular buffer, along with the associated pointers. The MPU's head pointer is labeled "HP", the tail pointer is labeled "TP", and the ASIC's read pointer is labeled "RP". In Figure 3a, five packets have been placed in memory for the ASIC to retrieve. Their addresses are in the circular buffer and the valid bits of those entries are set as a signal to the ASIC that data is available. In Figure 3b, three more packets have been written to the NetRAMs and four of the original packets have been retrieved by the ASIC. The ASIC has written a 0 to the valid bit of the first four entries to signal the MPU that the data has been retrieved. The MPU has not yet responded by pushing the addresses back onto the free buffer list. This is apparent because the head pointer is still pointing to the first address placed into the circular buffer. In Figure 3c, no new packets have been written to or read from the packet data memory. However, the MPU has reclaimed the first four locations in the circular buffer by pushing the addresses back onto the free buffer list and advancing the head pointer to the first entry with a valid bit of 1.

If the system needs to support the transfer of packets of different priorities, only the circular buffers need to be modified. Each buffer should be split into multiple, smaller buffers that hold the addresses of packets of a given priority. The receiving computing element can use strict priority, retrieving the highest priority packet that is available at any time. Alternately, it can use a weighted round robin algorithm that guarantees some bandwidth to the lower priority classes. In most cases, a single priority queue can be used for the return trip from ASIC to MPU, assuming the MPU has enough bandwidth to keep up with the ASIC.

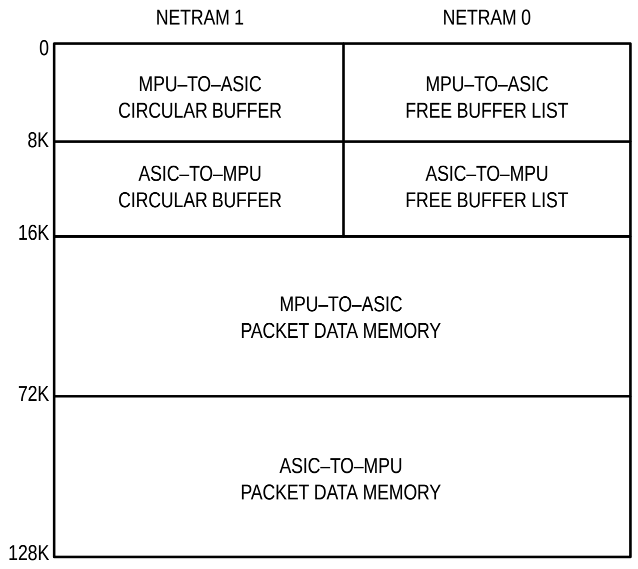
NETRAM 1
NETRAM 0
0
8K
16K
72K
128K
MPU–TO–ASIC CIRCULAR BUFFER
MPU–TO–ASIC FREE BUFFER LIST
ASIC–TO–MPU CIRCULAR BUFFER
ASIC–TO–MPU FREE BUFFER LIST
MPU–TO–ASIC PACKET DATA MEMORY
ASIC–TO–MPU PACKET DATA MEMORY

Figure 2. Memory Map

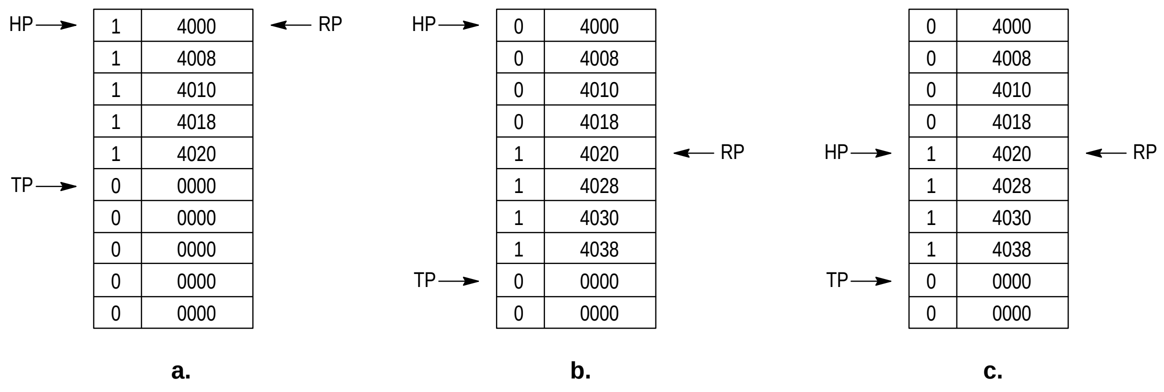

Figure 3. Circular Buffer Example

FREE BUFFER LIST DESCRIPTION

The system includes a separate free buffer list for each direction of traffic flow. These buffers are implemented as stacks in NetRAM 0, as shown in the memory map of Figure 2. The memory map also shows the 56K words from address 4000_{16} to $11FFF_{16}$ are assigned to the MPU–to–ASIC packet data memory, and the 56K words from address 12000_{16} to $1FFFF_{16}$ are assigned to the ASIC–to–MPU packet data memory. Since each packet comprises 64 bytes (or 8 words) of packet memory, the system initializes the free buffer lists by placing every eighth address of the respective ranges into the corresponding stack.

When a packet is available to be transferred to the ASIC, the MPU pops a value from the MPU–to–ASIC free buffer list. This value represents the address in packet memory to which the data should be written and the value that should be placed in the circular buffer along with a valid bit of 1. After the ASIC has read the packet, it will clear the valid bit in the circular buffer. When the MPU recognizes this action, it will push the address from this entry of the circular buffer back onto the free buffer list.

Note the MPU and the ASIC only manipulate their own free buffer list. As a result, the lists can be maintained in memory that is not part of the dual port RAM. If such a scheme can be used, the designer will maximize the memory that is available for transferring packets. Use of the free buffer lists may be eliminated completely in the simple, single priority system. If one bit of each packet can be reserved to serve as the valid bit, the dual port memory map can be split into MPU–to–ASIC packet data memory and ASIC–to–MPU packet data memory, with no need for free buffer lists or circular buffers. In this case, the receiving computing element simply waits for the valid bit to be set in packet data memory. When it is set, the receiving element retrieves the packet and clears the bit.

MPU ALGORITHM DESCRIPTION

The MPU is responsible for transferring unprocessed packets to the ASIC, retrieving processed packets, and maintaining its free buffer list and circular buffer. The steps of the algorithm are summarized in Figure 4a. The MPU begins by initializing the circular buffer to all 0 values, which serves to clear all the valid bits. The processor then initializes the free buffer list, writing values corresponding to every eighth address in the MPU–to–ASIC packet data memory. In this initialized state, the circular buffer head and tail pointers contain the address of the buffer's first entry. In addition, the MPU read pointer for the ASIC–to–MPU circular buffer points to the first location of that buffer.

When the MPU is ready to begin transferring data, it first checks for the availability of an unprocessed packet. If one is found, and a location is available in the packet data memory, the processor writes the packet to the designated address it retrieves from the free buffer list. It then adds the packet address, with a valid bit of 1, to the circular buffer. Finally, the MPU increments the circular buffer's tail pointer.

Next, the processor determines if the ASIC has read any of the packet data memory locations. It does this by reading from the head of the MPU–to–ASIC circular buffer. If the valid bit is cleared, this indicates the corresponding packet has been processed. The MPU pushes the address from this location of the circular buffer back onto the free buffer list, and increments the circular buffer's head pointer. This process is repeated until the MPU finds an unprocessed packet or determines that packet data memory is empty.

The MPU then checks for packets that are ready for the return trip from the ASIC. It checks the location of the ASIC–to–MPU circular buffer indicated by its read pointer, looking for a valid bit of 1. If the bit is set, it retrieves 8 words from the ASIC–to–MPU packet data memory beginning at the location specified in the circular buffer entry. After retrieving the data, the MPU clears the valid bit and increments its read pointer. The process of reading from the ASIC–to–MPU circular buffer, reading the corresponding packet, clearing the valid bit, and incrementing the read pointer is continued until a location is found in the circular buffer with a valid bit of 0.

At this point, the MPU is ready to begin a loop. It will continue to check for unprocessed packets, determine if the ASIC has retrieved any packets, and read any processed packets returned by the ASIC.

1. Initialize the MPU–to–ASIC circular buffer. (Write 0s to all locations.)	1. Initialize the ASIC–to–MPU circular buffer. (Write 0s to all locations.)
2. Initialize the MPU–to–ASIC free buffer list. (Write 4000_{16} – $11FF8_{16}$, incrementing by 8, to addresses 0000_{16} – $1BFF_{16}$ of NetRAM 0, and 0s to addresses $1C00_{16}$ – $1FFF_{16}$.)	2. Initialize the ASIC–to–MPU free buffer list. (Write 12000_{16} - $1FFF8_{16}$, incrementing by 8, to addresses 2000_{16} - $3BFF_{16}$ of NetRAM 0, and 0s to addresses $3C00_{16}$ - $3FFF_{16}$.)
3. If a packet is available for transfer to the ASIC and a location is available in the MPU–to–ASIC circular buffer, pop the MPU–to–ASIC free buffer list and write the packet to the designated address in the MPU–to–ASIC packet data memory. Add the packet address and a valid status bit to the end of the MPU–to–ASIC circular buffer. Increment the MPU–to–ASIC circular buffer tail pointer.	3. If a packet is available for transfer to the MPU and a location is available in the ASIC–to–MPU circular buffer, pop the ASIC–to–MPU free buffer list and write the packet to the designated address in the ASIC–to–MPU packet data memory. Add the packet address and a valid status bit to the end of the ASIC–to–MPU circular buffer. Increment the ASIC–to–MPU circular buffer tail pointer.
4. Read the head of the MPU–to–ASIC circular buffer. If the valid bit is cleared, push the address onto the MPU–to–ASIC free buffer list and increment the MPU–to–ASIC circular buffer head pointer. Repeat until an entry is found with the valid bit set or until the MPU–to–ASIC circular buffer head pointer equals the MPU–to–ASIC circular buffer tail pointer.	4. Read the head of the ASIC–to–MPU circular buffer. If the valid bit is cleared, push the address onto the ASIC–to–MPU free buffer list and increment the ASIC–to–MPU circular buffer head pointer. Repeat until an entry is found with the valid bit set or until the ASIC–to–MPU circular buffer head pointer equals the ASIC–to–MPU circular buffer tail pointer.
5. Read an entry from the ASIC–to–MPU circular buffer, using the MPU read pointer. If the valid bit is set, read the packet at the corresponding address in the ASIC–to–MPU packet data memory. Clear the valid bit without modifying the address. Increment the MPU read pointer. Repeat this entire step until an entry with a cleared valid bit is found.	5. Read an entry from the MPU–to–ASIC circular buffer, using the ASIC read pointer. If the valid bit is set, read the packet at the corresponding address in the MPU–to–ASIC packet data memory. Clear the valid bit without modifying the address. Increment the ASIC read pointer. Repeat this entire step until an entry with a cleared valid bit is found.
6. Check for more data that needs to be sent to the ASIC.	6. Process any packets that were received from the MPU.
7. Go to step 3.	7. Go to step 3.
a. MPU Algorithm	**b. ASIC Algorithm**

Figure 4. System Algorithms

ASIC ALGORITHM DESCRIPTION

The ASIC's responsibilities are very similar to those of the MPU. It transfers processed packets back to the MPU, retrieves packets the MPU has stored in the NetRAM for it, and maintains its free buffer list and circular buffer. Like the processor, it begins by initializing its circular buffer, free buffer list, and read pointer. Figure 4b summarizes the steps in the ASIC algorithm.

When the ASIC is ready to begin transferring data, it first checks for the availability of a processed packet. If one is available, and a location is free in the packet data memory, the ASIC writes the packet to the designated address it retrieves from the free buffer list. It then adds the packet address, with a valid bit of 1, to the circular buffer. Finally, the ASIC increments the circular buffer's tail pointer.

Next, the ASIC determines if the MPU has read any of the packet data memory locations. It does this by reading from the head of the ASIC–to–MPU circular buffer. If the valid bit is cleared, this indicates the corresponding packet has been processed. The ASIC pushes the address from this location of the circular buffer back onto the free buffer list, and increments the circular buffer's head pointer. This process is repeated until the ASIC finds an unprocessed packet or determines that packet data memory is empty.

The ASIC then checks for cells that have been written into packet data memory by the MPU. It checks the location of the MPU–to–ASIC circular buffer indicated by its read pointer, looking for a valid bit of 1. If the bit is set, it retrieves 8 words from the MPU–to–ASIC packet data memory beginning at the location specified in the circular buffer entry. After retrieving the data, the ASIC clears the valid bit and increments its read pointer. The process of reading from the MPU–to–ASIC circular buffer, reading the corresponding packet, clearing the valid bit, and incrementing the read pointer is continued until a location is found in the circular buffer with a valid bit of 0.

At this point, the ASIC goes into its loop. It will continue to check for processed packets, determine if the MPU has retrieved any packets, and read any unprocessed packets written by the MPU.

SUMMARY

This application note describes a solution to a problem commonly encountered in the design of data communications equipment. The NetRAM dual port memory products provide an efficient means of transferring packets between two computing elements in a switch or router. The NetRAMs also support the circular buffer handshaking mechanism that coordinates the flow of data between the two processors. While a single priority algorithm is described, the circular buffers are easily extended to support multiple priorities of traffic flowing between the MPU and ASIC.

NOTES

NOTES

NOTES

How to reach us:

USA / EUROPE / Locations Not Listed: Motorola Literature Distribution; P.O. Box 5405, Denver, Colorado, 80217. 1-303-675-2140 or 1-800-441-2447

Mfax™**:** RMFAX0@email.sps.mot.com – TOUCHTONE 1-602-244-6609
Motorola Fax Back System – US & Canada ONLY 1-800-774-1848
– http://sps.motorola.com/mfax/

HOME PAGE: http://motorola.com/sps/

JAPAN: Motorola Japan Ltd.; SPD, Strategic Planning Office, 141, 4-32-1 Nishi-Gotanda, Shinagawa-ku, Tokyo, Japan. 81-3-5487-8488

ASIA / PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Centre, 2 Dai King Street, Tai Po Industrial Estate, Tao Po, N.T., Hong Kong. 852-26668334

CUSTOMER FOCUS CENTER: 1-800-521-6274

MOTOROLA