```
+---------------+
! d i g i t a l !   i n t e r o f f i c e   m e m o r a n d u m
+---------------+
```

To: Dolphin Project List

Date: 7 August 1978

From: John Allen [illegible]
Dept: L.C.E.G.
Tel: 231-6713
Loc: MR1-2/E47
File: MBFUN
Rev: 1

Dolphin Mbox Functional Specification Revision 1

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-10
DECCOM	DECsystem-20	TYPESET-11

1.0 INTRODUCTION

The Dolphin central processor consists of three or four major subsystems. They are the Ebox, which performs the traditional tasks of an arithmetic central processor, the Ibox, which prefetches and decodes instructions for the Ebox, the Mbox, which fetches data from main memory and caches it for the above units and for the optional FPA (Floating point accelerator), which provides high speed arithmetic processing of floating point operands in parallel with the Ebox. The Ebox, Ibox and FPA combined are called the "Pbox"

The Dolphin Mbox is composed of four principal functional units, the microcontroller, the data cache, the page table cache, and the bus interface.

2.0 TECHNOLOGY

The Dolphin Mbox will occupy two Extended Hex multilayer etch modules. It will use ECL 256 x 4 bit memories and Macro cell array LSI logic circuitry. There will be approximately 35 Macro cell arrays and 162 memories including the Bus interface. (There are 10 Macro cell array part types including the Bus and ECC parts.) A small number of small scale integration parts will be used, primarily as buffers and parity checkers.

3.0 MBOX FUNCTIONS AND FEATURES

1. 2048 word cache of Pbox instructions and read and write data.

2. Virtual address translation by a 512 entry page table cache.

3. Control and interface to the Dolphin Bus for interrupts and memory requests.

4. Error recovery using ECC and special hardware and software on all memories and interboard bus signals.

5. A microcontroller to handle bus operations, cache refills, cache sweeps, and word invalidates, and other operations traditionally performed by hard wired logic.

6. High performance symmetrical processor support is designed in. Multiple processors' caches are automatically kept up to date with the optional shared pages data integrity box which plugs into the Dolphin bus.

4.Ø PRINCIPAL OF OPERATION

4.1 Overview

The Mbox combines a microcoded controller with small amounts of control logic to realize high speed with a minimum of unchecked control logic. This is an extension of the philosophy of design of the KL1Ø Mbox. In the KL1Ø, only the paging operations for KL paging were controlled by the Ebox microcode. In the Dolphin Mbox, all paging is controlled by the Ebox. Cache sweep, writeback operations and Dolphin bus control operations are controlled by the Mbox micro controller.

4.2 General Description

The Dolphin Mbox is very similar in function to the KL1Ø Mbox. In particular, it contains a "writeback" cache of 128 X 4 X 4 words, keyed by physical address. The physical address size has been expanded from 22 bits to 27 bits, necessitating an increase in the size of the cache directory from 13 to 18 bits of physical page number. An autonomous cache sweeper is included, which will be invoked by the same instruction as in the KL1Ø (unless someone objects).

A paging cache is included which is organized as 512 X 1 X 1, increasing the Page table cache hit rate over the KL1Ø's 128 X 4 X 1. Page table cache refill time is reduced (from the KL1Ø) with the use of a cache of section pointer entries. The section pointer cache will be accessed and controlled by the Ebox microcode and it will use part of the Ebox scratch memory. Since the virtual address has expanded to 3Ø bits plus "User", the directory for the paging cache, plus the physical page entry now total 36 bits. The lookups in these two caches happen simultaneously, as in the KL1Ø. No paging algorithm is implemented in the Mbox, and all paging cache misses or access control violations are reported to the EBOX. As in TOPS-2Ø paging on the KL1Ø, the Ebox must be able to follow the appropriate (TOPS-1Ø or TOPS-2Ø) algorithm, and write the paging cache, without restarting the instruction.

In accordance with the principles agreed on at the 16 June "Brainstorming" session (see J. Allen memo "MBOX4" and A. Kotok memo "SHARED2" of 2Ø July 1978), the Mbox will accept bus requests to invalidate cache entries. The exact scenario of how interlocks (AOS, etc.) are handled may be found in these memos.

The Mbox will handle the Dolphin bus related part of the processor PI system. It will interrupt the Ebox when a PI request arrives on the bus and it will handle sending out "broadcast polling messages" and responding to device responses to the poll.

4.3 Detailed Description

The Mbox will incorporate a fast, pipelined micro controller which will honor requests for major cycles from the Ebox and Ibox, as well as internally generated requests for cache sweeping and bus-originated cache zap requests. A major cycle is defined as a flexible period of time during which the "VMA bus" is driven from a single source. In a typical operation, the Ebox would request a cycle one tick (16.67 ns) ahead of when its VMA register will be loaded. The Mbox controller would issue a grant signal, "MBOX GRANT" which will cause the Ebox VMA to be gated onto the VMA bus, and the cycle-type information from the Ebox to be examined. Assuming a virtual read request was issued, the controller would allow adequate time for the various paging and cache comparisons, and ECC checks to stabilize. At such time, if all conditions were successful, the data and check bits would appear on the Mbox output lines, and a signal would be supplied to the Ebox indicating it can proceed. During the last tick of the major cycle, arbitration would take place to award the next cycle.

Various conditions might interfere with this happy scenario. If the desired word were not in the cache, the Mbox controller would retreive it from main memory. Depending on timing, the Mbox may always issue a Bus request in speculation that it may be required, and then not use the cycle if not needed. Memory requests will in general be 4 word requests, as in the KL10. If some of the words are not needed in the cache, the Mbox will discard them. During the time the Mbox is waiting for memory to respond, it will only accept directed cycles to it from the memory, and be prepared to acknowledge and discard certain PI polling responses. All other cycles will be "busy acknowledged". While waiting, the Mbox controller will write the appropriate entries in the cache directory, and the valid and written bits. When this is done, the Ebox response signal will be held up, and the Mbox controller will just count a timeout.

When the data arrives from the memory, the desired word will be put on the Mbox output lines, and upon indication that it was accepted by the Ebox (or possibly without such indication), the Mbox controller will proceed to write the received words into the cache. The desired word will be held on the "MBOX DATA" output lines at least until the Ebox asserts "Ebox RECEIVED DATA". [More to come.]

5.0 DETAILED PBOX INTERFACE DESCRIPTIONS

5.1 Pbox Virtual References; Reads And Writes:

The following are the signals passed between the Ebox/Ibox/FPA and the Mbox for Virtual (paged) read and write references. The Ebox is the only device of the three that does any operation other than memory reads.

* Note: physical references use the same interface signals as the virtual reference signals with the exception of the items marked above with an asterisk.

1. * Ebox - "PHYS REF" (physical reference) false.

2. * Pbox - "VMA BUS" as per figure 5.1. These enable checking of sub parts of the VMA BUS.

3. Pbox - VMA bus parity - several parity bits over parts of the field.

4. Pbox - "READ" - Asserted on or before VMA and Mbox request.

5. Ebox - "WRITE" - Asserted on or before VMA and Mbox request.

6. * Ebox - "WRITE TEST" - Mbox will test legality of a write before doing any function. This signal is guaranteed to only be asserted on virtual references. The Mbox will do a write test if and only if this signal is asserted.

7. Pbox - "MBOX REQUEST" - This will be asserted not more than one Mbox clock tick before the VMA is valid. Request type signals such as "PHYS REF", "READ", "INTERLOCK", or "WRITE TEST" will be asserted on or before Mbox request time.

   Tentatively, the Mbox may begin processing a request before the previous request has been accepted by the Pbox by asserting "EBOX RECEIVED DATA". The Mbox output data will not be affected by this.

8. Ebox - "EBOX DATA" - 36 bits of data plus ECC check bits, 43 bits total. It must be valid at most one Mbox tick after "MBOX REQUEST".

9. Pbox - "LOOK" - Look in the cache for this reference. (If a virtual reference, the Mbox ands "LOOK" with PT cacheable.)

10. Pbox - "LOAD" - Load new data into the cache. Look must be true. (If a virtual reference, the Mbox ands "LOOK" and "LOAD" with PT cacheable.)

11. Pbox - "WRITETHROUGH" - Write into both Main memory and the cache. "LOAD" must be true. (If a virtual reference, the Mbox ands it with (PT cacheable or PT writethrough.)

12. Pbox - "PBOX RECEIVED DATA" - This signal acknowledges data and clears "MBOX RESPONSE".

13. Pbox - "INTERLOCK" - Requests are only interlocked if "INTERLOCK" is true and

    1. "LOOK" and "LOAD" = Ø or

    2. "PT CACHE" = Ø or

    3. "PT WRITEBACK" = 1.

14. Pbox - "PBOX BACKGROUND READ" -Similar to Pbox read except that it must be followed up by a regular request before the Mbox will respond with a "MBOX RESPONSE". Also, the Mbox will abort the servicing of the request it any other request comes along and the Mbox will never request main memory solely on the basis of a "PBOX BACKGROUND READ".

    If "PBOX BACKGROUND READ" was seen by the Mbox three or more ticks before a regular read request and the Mbox had not aborted it, and the page table cache entry was valid, and the data was in the cache, then on the clock tick after the regular request, "MBOX RESPONSE" will be asserted and the data will be available on the "MBOX OUT" lines.

    This provides a means for the Pbox to maximize utilization of the Mbox without some of the serious contention problems that would otherwise result.

15. Ebox - "WRITE DATA VALID" - Asserted by Ebox when the data being written is available at the Ebox Data Out Lines.

16. Pbox - "ABORT MBOX CYCLE" (As in KL1Ø AC Ref) This may be asserted at almost any time during any cycle. Exception cases will be specified later. It will also be used by the Ebox to clear a page fail state.

17. Mbox - "MBOX GRANT" - There is one Mbox Grant signal line for each device which may access the VMA BUS. For example, "MBOX GRANT IBOX". It enables the highest priority requestor of the Mbox to access the VMA bus. Bus data is read onto the VMA bus at the start of the next Mbox clock.

18. Mbox - "MBOX DATA" - 36 bits of data plus ECC check bits, 43 bits total. Valid when "MBOX RESPONSE" true.

19. Mbox - "MBOX RESPONSE" - On write; Mbox received write data. On read; Data available at output latch.

20. Mbox - "PAGE FAIL" - This signal is the catch all for most exception conditions including page table access failures, processor interrupts, ECC errors and NXM traps. It will stay asserted until the Ebox asserts "ABORT MBOX CYCLE". It is enabled by "Mbox request" so the Ebox must do an "ABORT MBOX CYCLE" before it's first Mbox request after the page fail or else the Ebox will page fail again. The following conditions result in page fail:

    1. * Page Table cache no match.

    2. * Page table write access failure.

    3. * Page table Written state transition.

    4. Processor automatic priority interrupt. As in the KS10, instead of having the microcode check for interrupts, the Mbox will page fail the microcode when next it does an Mbox request.

    5. ECC or parity error.

    6. NXM (Non-existant memory) error.

    7. Incomplete memory cycle.

21. Mbox - Five or six bit problem type code that indicates the type of failure for the above sources of page failure. This problem type code will be used by the Ebox microcode to dispatch to the appropriate service routine. References which receive a page fail will not get a Mbox response.

Figure 5.1

VMA bus Virtual Reference Format

```
 Ø   3 4 5 6              17 18          26 27          35
!-----------------------------------------------------------!
!Must ! ! !     Virtual     !  Virtual    !  Virtual    !
! be  !Ø!U!     Section     !   Page      !   Line      !
!zero ! ! !                 !             !             !
!-----------------------------------------------------------!
```

Note: on a virtual reference, the Mbox will generate a zero byte mask. Also, the Mbox must provide a write test function for the Ebox/Ibox.

5.2 Ebox Physical References; Reads/writes:

* Note: physical references use the same interface signals as the virtual reference signals with the exception of the items marked above with an asterisk.

All others above plus the following apply.

1. Ebox - "Physical Reference" true

2. Ebox - VMA bus as per figure 5.2 below.

Figure 5.2

VMA bus Physical Reference Format

```
 0    3 4    7 8 9            17 18           26 27           35
!----------------------------------------------------------------!
!Must !     !I!                                                  !
! be  ! Mask!/!             Physical Address                     !
!zero !     !O!                                                  !
!----------------------------------------------------------------!
```

5.3 Ebox Page Table And PT Directory Read Special Function.

Page table read supplies the page table address on bits 18 to 26 of the VMA BUS. Both the page table and the page table directory data are returned on the MBOX DATA linesin the format specified by figure 5.3 below.

Figure 5.3

Page Table Cache Read/Write Format

```
 0     4 5 6 8 9                          26 27        35
!--------------------------------------------------------!
! Access!U! PT!  Physical page number     !   Page      !
! state !S!Dir!                           !   table     !
! bits  !E!6-8!       9 - 26              ! directory   !
!       !R!   !                           !   9 - 17    !
!--------------------------------------------------------!
```

5.4 Ebox Page Table And PT Directory Write Special Function.

The Ebox supplies the page table address on bits 18 to 26 of the VMA BUS. The data for the specified page table entry and the page table directory entry is supplied on the EBOX DATA OUT lines with the format of figure 5.3 above.

5.5 Sweep Functions

Sweep special functions will consist of a signal "SWEEP REFERENCE" presented to the Mbox with two or more qualifiers; one from group 1 below and one or both from group 2 below:

Group 1 -

1. Sweep one word
2. Sweep one four word block
3. Sweep one page
4. Sweep all pages

Group 2 -

1. Invalidate cache
2. Validate memory

5.6 PI System

NOTE: As this spec is being written, the PI system design is evolving. Therefore, the following information may significantly change.

The PI system in the Dolphin is distributed due to the nature of the unified bus design. Because of this unification, the Mbox, which is the interface to the Dolphin bus, must handle all control type bus transactions. This includes interrupts and the Ebox's handling thereof which were previously handled by the KL1Ø Ebus.

The Mbox will provide the Ebox with the capability of reading and writing the bus "ID" field and the bus "TAG" field. It will also allow the Ebox to get a bus cycle similar to the KL1Ø's "Get ECL Ebus". The Ebox will be able to cause the Mbox to generate a "Broadcast" function and the Mbox will handle the responses to these broadcast functions. This involves saving the ID of the first interrupting device that responds to the Broadcast function and also comparing the ID of an interrupting device to a saved ID field.

When an interrupt or other message for the CPU is received on the bus, the Mbox will page fail the Ebox. [More detail later on this.]

The Ebox will be able to read the PI level (on bits 33-35 of "MBOX DATA").

6.0 DETAILED BUS INTERFACE DESCRIPTION

The Mbox will use the standard 11 MCA Dolphin bus interface chip set. The Mbox data path will provide the receive buffer space. The Mbox microcontroller will initiate all bus operations and handle all incoming bus requests and data returns. For more information, see the Dolphin Bus Specification.

7.0 DETAILED CACHE FUNCTIONAL DESCRIPTION

To be supplied. See KL10 hardware maintenance manual. The Cache will use the same structure and algorithms as the KL10 cache. State control will be provided by the Mbox microcontroller. (MNA chip). MCAs involved in the cache control are the CSH, VAW and MAD chips plus the microcontroller.

7.1 Cache Sweeps

Cache sweeps will be done by the Mbox microcontroller. They will be significantly faster than the KL10 and will continue to be able to run asynchronously to and in parallel with the Pbox. [See interface sections for more information.]

7.2 Cache Writeback/refill

Cache writeback/refill will be controlled by the Mbox microcontroller.

7.3 Cache Invalidate Word/block Per Bus Request. (Cache Zap).

Cache invalidate functions will be controlled by the Mbox microcontroller, which will handle commands from the bus interface and service them by clearing valid and written bits in one to four words of cache directory.

8.0 DETAILED PAGE TABLE FUNCTIONAL DESCRIPTION

All of the required state information for access of a page will be encoded into three bits. This includes the states involving the access, writeable, modified, cacheable and writeback conditions in the PT directory. They were to be encoded into three bits (eight states total) as follows:

1. Page Not in hardware PT (not valid)
2. Not cacheable, not writeable
3. Not cacheable, writeable, not written
4. Not cacheable, writeable, written
5. Cacheable, not writeable
6. Cacheable, writeable, not written
7. Cacheable, writeable, written
8. Writethrough (= Cacheable, writeable, written and Writethrough)

A fourth bit which is independent of the above states, would indicate that a CST update is needed on next refill of the page table.

[More Details to be supplied.]

9.0 FUNCTIONS NOT PERFORMED BY THE MBOX

9.1 All Handling Of The UBR And EBR.

The UBR and EBR are kept in a scratch pad by the Ebox. It is solely responsible for their maintenance.

9.2 The Map Instruction.

The Map instruction will be implemented in the microcode by tracing through page tables in memory. There is no Mbox assistance and therefore, it will be slower than in the KL10.

9.3 Sbus Diag Functions

All functions currently equivalent to "Sbus Diag" become Dolphin bus I/O reads and writes.

10.0 PERFORMANCE

10.1 Overall Goal And Committment

The performance of the Mbox in terms of overall system throughput is almost solely a function of the cache access time. (The second order effect is Main memory access time and a third order effect is the pager refill time. The goal for the Dolphin Mbox is for a cache access time of 50 nanoseconds. This yields a most likely logic mix performance of 2.1 times a 2050 at 93% cache hit rate.

10.2 Page Table

The page table hit rate will be somewhat better than the KL10 for an equivalent software environment in that there are 512 directory entries versus 128 in the KL10. This will somewhat counteract the effects of the increased number of sections used by software.

10.3 Section Pointer Cache

The section pointer cache is provided and controlled by the Ebox microcode using the Ebox's scratch RAM. [See Ebox functional specification.]

11.0 R.A.M.P. FEATURES

11.1 Single Error Correction ("ECC")

No single RAM error will be able to crash the system.

Almost all Random access memories, address and data paths carry ECC. The exceptions are the VMA bus, which will carry parity, the use bits and history bits, which will have parity and which can not cause fatal errors to the system, only lower cache hit rates, and the Valid and Written bits. The Valid and Written bits will have multiple error correction provided that the operating system can tolerate the writing back to memory from the cache of a word that had not been written by the processor since it was last read into the cache. If the operating system can not tolerate the writeing back of a non-written word, a written bit single error will be handled by the monitor similarly to current parity errors - recover, crash user, or crash system depending on the situation. In almost all cases, the error will be non-fatal. Unchecked random control logic will be held to the barest minimum amount possible. [See the Dolphin RAMP plan for more information]

11.2 Error Correction Method

Due to the need to use data before the error can be detected, the correction method will be to retry the reference with slower clocking to allow time for the syndrome correction to propagate through the longer path.

11.3 Error Logging

All correctable and uncorrectable errors will be reported to the Ebox microcode by means of an page fail.

Uncorrectable errors may temporarily stop the Mbox clock after latching information on the failure and interrupting the maintenance console. (This would not affect the Memory clocking.) The console will poll the diagnostic logic to find out the failure type and gather all possible state data and then will continue or restart the Mbox. Error state information will be latched in registers where applicable. [More to come.]

I propose that the Ebox microcode have a "logout" area in main memory where it may put the information for the use of the monitor. The Mbox may have functions which allow the microcode to disable the interrupt for correctable errors. (Or the Ebox may be told by the operating system to ignore them.) This will reduce the amount of repetitive information in the system error file and reduce error handling overhead.

The Mbox microcode may implement error retry for selected fault conditions (other than correctable ECC errors).

12.Ø COST

To be supplied - See Vic Ku's system cost breakdown.

13.Ø EXPANDABILITY

13.1 Cache

The cache may be expandable from the minimum 2K to 4K or 8K words if there is significant performance improvement with resonable differential performance change. Present calculations indicate a logic performance improvement of less than 5% for a 2.5% increase in manufacturing cost. This is based on doubling the cache size using the same RAMs. (There are two problems with cache expansion - either the page table must be cycled in series with the cache to provide more address bits for the cache (this may require a faster 256 bit memory), or a greater level of associativity must be provide which makes the implementation of a Least Recently Used refill algorithm much more difficult. Use of simpler algorithms such as pseudo-random has not been investigated.)

13.2 Physical Address Space

No effort will be made to make the physical address space expandable.

13.3 Multiple Processors

The goal is to allow configuration of up to four processors under both TOPS-1Ø and TOPS-2Ø.

A scheme to allow cacheing of shared pages by multiple processors will be implemented. The scheme will provide automatic clearing of invalid data from the caches of multiple processors. For more detailed information on this scheme, see Alan Kotok's memo of 2Ø July 1978, "Shared Pagess in Multiprocessor Systems" (SHARED2.MEM).