VME BUS PERFORMANCE IN MULTIPROCESSOR SYSTEMS

Glenn Mayer & Duane Voy

Fermi National Accelerator Laboratory[1]
P.O. Box 500 Batavia, Illinois 60510

Abstract

The complexity of future control systems will require large amounts of processing power in each individual subsystem. This can be achieved by using a higher performance processor or by using multiple processors within a single system. Multiple processors on a single bus may saturate the bus so as to reduce the expected performance gain. In order to determine an empirical limit for the number of processors that a VME bus can support, experiments were conducted to find the point at which VME system output was not increased by the addition of more processors. This paper details the results of those tests.

Goals of the Test

In order to generate some facts[2] about the performance of VME in a multiprocessor configuration, a simple set of tests were conducted. The goals of these tests were to:

- Determine the incremental system performance improvement as a function of the number of CPUs
- Determine how much effect different arbitration schemes have on system performance
- Determine the system performance at various bus utilization levels

Test Hardware Configuration

The test setup hardware was assembled from available components. These are not high performance parts but are representative of the hardware that is used to build embedded systems at Fermilab. The test hardware consisted of:

1 VME Crate with power supply
1 Bus Arbiter Board (Motorola MVME025)
5 16Mhz 68020 CPU Boards (Motorola MVME133-1)
1 Memory Board (Micro Memory MM-6700)
1 Crate Utility Board (Fermilab)
1 Oscilloscope

[1]Operated by Universities Research Association, Inc. under Contract with the United States Department of Energy.

[2]If it can't be expressed in numbers it's opinion, not fact.

Bus Arbitration

Each of these tests was run with three different CPU configurations. The first configuration had one CPU board on each of the four different VME bus priority levels. The bus arbiter was configured for Priority Arbitration (PA) giving CPU 1 highest priority, CPU 2 next highest, etc. The second configuration was the same as the first except that the bus arbiter was configured for Round Robin Arbitration (RRA). The last configuration had CPU 1 on bus priority level 3, and CPUs 2 through 5 on bus priority level 2 with geographical priority within the group. This configuration was intended to simulate a typical system with Mixed Arbitration (MA).

The M133-1 board [1] only releases the bus upon request (ROR). A solitary bus master does not incur arbitration overhead and therefore runs very efficiently. In order to plot the data in numbers that are representative of a multiprocessor environment, the amount of work that one CPU board can do when in a two-processor configuration was defined to be "one CPU worth of work". All of the following plots have the y-axis normalized to one CPU worth of work. The reasoning behind this normalization is that the initial drop in performance from one to two processors is due to arbitration overhead, not lack of bus bandwidth.

Software

The following basic test algorithm was executed on each CPU board:

- Activate a CPU specific test signal on the crate utility board
- Write 2K words of data to the memory board
- Read the 2K words back from the memory board
- Repeat

Performance was determined by measuring the time required to execute the above loop. Two variations of the basic test algorithm provided different levels of bus utilization. The first variation was a 2 Instruction Loop (2IL) designed to produce worst case bus traffic (See Figure 1). The second variation was an 8 Instruction Loop (8IL) memory test algorithm that was designed to produce more "realistic" bus traffic (See Figure 2). The instruction cache was not enabled during these tests.

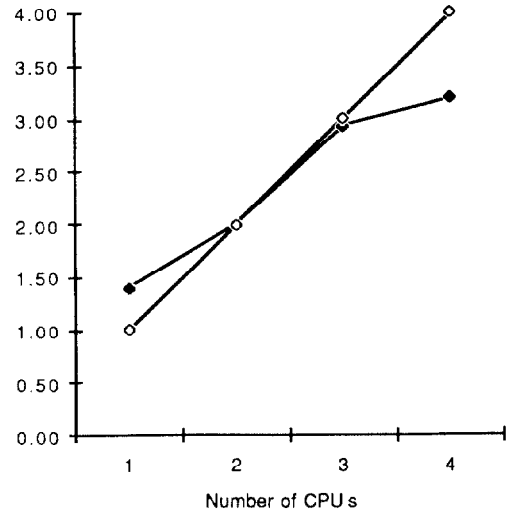
CH2669-0/89/0000-1654$01.00©1989 IEEE

```
        LEA     MEM_START,A0            * POINT TO TEST MEMORY
        MOVE.L  #MEM_COUNT/2,D3         * GET THE TEST COUNT
GENPAT:
        MOVE.W  D0,(A0)+                * SET THE TEST LOCATION
        DBRA    D3,GENPAT
        LEA     MEM_START,A0            * POINT TO TEST MEMORY
        MOVE.L  #MEM_COUNT/2,D3         * GET THE TEST COUNT
CHKPAT:
        MOVE.W  (A0)+,D6
        DBRA    D3,CHKPAT
```

Figure 1. Source code for 2 instruction loop.

```
        LEA     MEM_START,A0        * POINT TO THE TEST MEMORY
        MOVE.L  #MEM_COUNT/2,D3     * GET THE TEST COUNT
GENPAT:
        MOVE.W  D1,D0               * START WITH PATTERN MODIFIER
        MOVE.L  A0,D2               * ADD IN THE ADDRESS BITS
        EOR.W   D2,D0
        SWAP    D2
        EOR.W   D2,D0
        MOVE.W  D0,(A0)+            * SET THE TEST LOCATION
        SUBQ.L  #1,D3               * LOOP UNTIL MEMORY FILLED
        BNE     GENPAT
        LEA     MEM_START,A0        * POINT TO THE TEST MEMORY
        MOVE.L  #MEM_COUNT/2,D3     * GET THE TEST COUNT
CHKPAT:
        MOVE.W  D1,D0               * BUILD TEST PATTERN AS ABOVE
        MOVE.L  A0,D2
        EOR.W   D2,D0
        SWAP    D2
        EOR.W   D2,D0
        MOVE.W  (A0)+,D6
        CMP.W   D6,D0               * CHECK AGAINST TEST LOCATION
        BEQ     CHK1                * IF OK SKIP
        ADDQ.B  #1,ERROR_FLAG       * COUNT UP THE ERRORS
CHK1:
        SUBQ.L  #1,D3               * LOOP UNTIL MEMORY CHECKED
        BNE     CHKPAT
```

Figure 2. Source code for 8 instruction loop.



Raw Data (mS)

CPU 1	CPU 2	CPU 3	CPU 4	CPU 5	Total Work
5.47					1.40
7.64	7.64				2.00
7.77	7.80	7.90			2.93
7.90	8.21	9.18	16.31		3.20

Figure 3. VME Multiprocessor Test 2IL/PA.

Results

The following graphs (Figures 3-8) plot the amount of work performed vs the number of CPU's doing the work. The plots with hollow dots indicate the ideal of obtaining one CPU worth of work for every CPU added. The plot with the solid dots shows the measured amount of work performed for each additional CPU.

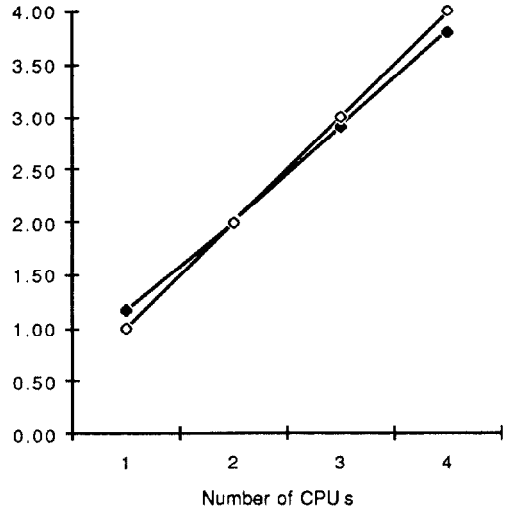
Caveats

It should be noted that this test measured the performance of a VME system built with components that are currently in use at Fermilab. Other CPU boards or arbiter boards may produce radically different results. In the above test, the system seemed to saturate at about $2X10^6$ bus cycles per second. The VME bus specifications [2] indicate that the bus should be capable of supporting more traffic. Those interested in using VME for a bus intensive system would need to find the performance bottlenecks and attempt to eliminate them.

References

[1] MVME133-1 VMEmodule 32-BIt Monoboard Microcomputer User Manual, 1987, Motorola Inc.

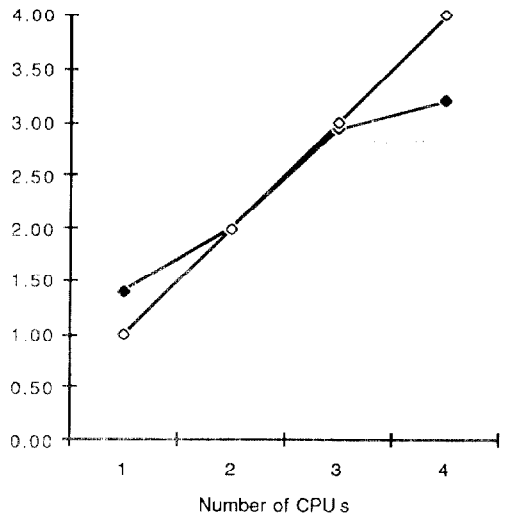
[2] The VMEbus Specification, Rev C.1, Oct. 85, Motorola, Inc.



Raw Data (mS)

CPU 1	CPU 2	CPU 3	CPU 4	CPU 5	Total Work
11.34					1.17
13.23	13.14				2.01
13.54	13.65	13.76			2.91
13.63	13.77	13.97	14.20		3.81

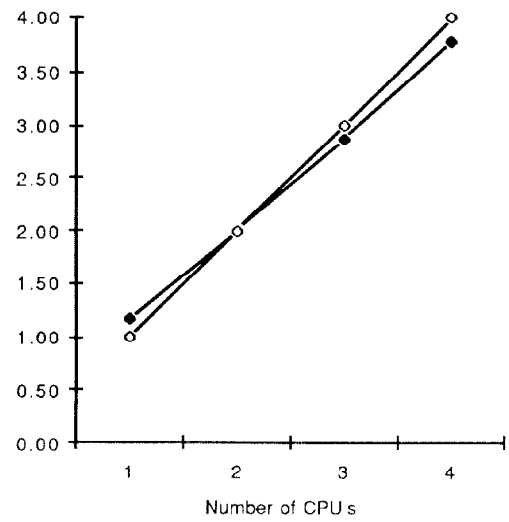
Figure 4. VME Multiprocessor Test 8IL/PA.



Raw Data (mS)

CPU 1	CPU 2	CPU 3	CPU 4	CPU 5	Total Work
5.52					1.40
7.71	7.71				2.00
7.87	7.87	7.87			2.94
9.61	9.63	9.63	9.63		3.20

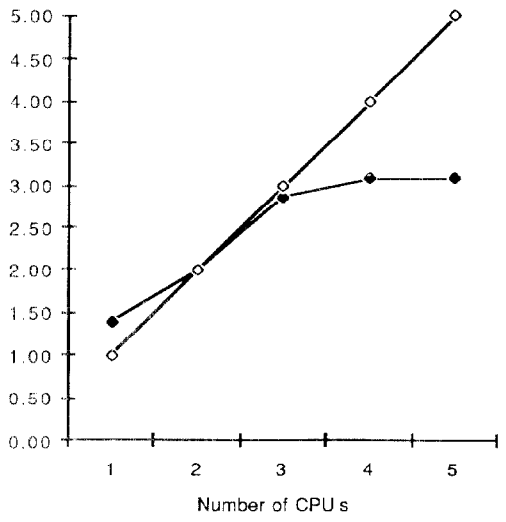
Figure 5. VME Multiprocessor Test 2IL/RRA.



Raw Data (mS)

CPU 1	CPU 2	CPU 3	CPU 4	CPU 5	Total Work
11.31					1.17
13.19	13.15				2.00
13.77	13.77	13.77			2.87
13.96	13.96	13.96	13.96		3.78

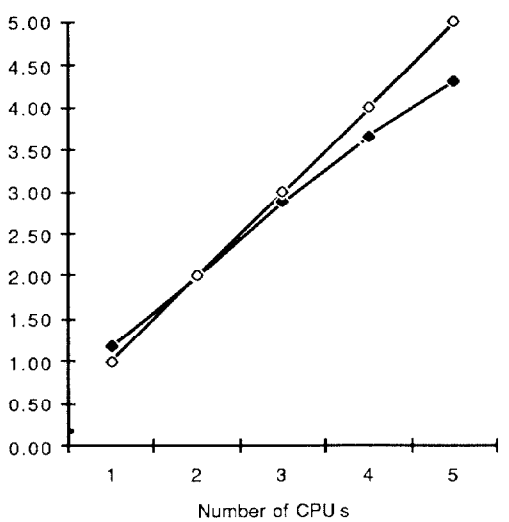
Figure 6. VME Multiprocessor Test 8IL/RRA.



Raw Data (mS)

CPU 1	CPU 2	CPU 3	CPU 4	CPU 5	Total Work
5.51					1.40
7.70	7.70				2.00
7.96	8.02	8.22			2.86
7.69	7.72	8.36	44.7		3.09
7.69	7.72	8.36	44.7	∞	3.09

Figure 7. VME Multiprocessor Test 2IL/MA.



Raw Data (mS)

CPU 1	CPU 2	CPU 3	CPU 4	CPU 5	Total Work
11.31					1.17
13.25	13.49				1.98
13.68	13.68	14.07			2.88
13.80	13.91	14.33	15.97		3.67
14.03	14.05	14.57	16.39	18.81	4.31

Figure 8. VME Multiprocessor Test 8IL/MA.