UTILIZING OFF-THE-SHELF MICROPROCESSORS FOR COMPLEX TELEMETRY PREPROCESSING

Christopher V. Ham
Systems Engineer
Physical Science Laboratory
New Mexico State University
P. 0. Box 3548
Las Cruces, New Mexico 33003-3548

ABSTRACT

This paper describes a system utilizing off-the-shelf microprocessor hardware to perform complex high-speed telemetry data preprocessing. The microprocessor equipment involves the latest in the Motorola computer series, namely the 68020 line. The author develops the specifications leading to the need of this type of preprocessor which is currently being developed under a contract to the McDonnell Douglas Helicopter Company. The paper fully describes the configuration of the hardware as well as the software available on the system. Detailed benchmarks of complex algorithms and other data manipulations are described. Test results relating parameter capacity and throughput are addressed. System architecture is described with the various trade-off analyses well defined. This system advances the art of preprocessing telemetry parameters requiring such functions as wild pointing, phase alignment concatenation, and derivations, all at rates in the megaword input.

INTRODUCTION

Until recently, high-speed telemetry systems have been limited as to the amount of data which could be processed in real time. Large amounts of raw data could be gathered and saved, but the actual processing had to he done on recalled rather than real-time samples. The Physical Science Laboratory, under contract to the McDonnell Douglas Helicopter Company is currently developing a telemetry preprocessing system which is capable of collecting and processing real-time data at the megaword-per-second rate.

The design goals of this system include 100 percent real-time processing of data, sustaining two million samples-per-second throughput and expansion capabilities for additional processing power. The processing mix includes conversion to Engineering Units (EU), identification of Wild Points (WP), and determining the average, maximum, and minimum values of n samples of data used in calculating resultants, computation of up to 5th order polynomials, and other user-defined equations using transcendentals and logarithmics.

The Motorola 68020 32-bit microprocessor was selected because it offered speed (16 MHz), large addressing range (4 gigabytes), extensive instruction set, floating-point support with the 68881 coprocessor, and the availability of software development tools. 68020 boards, along with memory and interface modules, are available off-the-shelf and are VMEbus-compatible.

HARDWARE CONFIGURATION

The preprocessing system consists of a set of Intermediate Processing Units (IPUs). These IPUs are VMEbus-based computer systems which contain 2 megabytes of Dynamic RAM for storage of EU tables, 256k bytes of Static RAM for program storage, and two 68020/68831 computer modules. Each of these boards is dual-ported to allow for maximum processing power and speed. Furthermore, VMEbus modules are available for interfacing an IPU to other systems, including VAX and SEL computers. Figure 1 shows an IPU configuration.

The dual CPU approach allows for one processor to perform the EU, WP, and other statistical processes, while the second calculates the polynomials and user-defined algorithms used to derive resultant values. Since both CPU modules contain a coprocessor, complex mathematical functions are performed in parallel with other processes. Interface boards reside on the VMEbus and can be accessed by either CPU.

The system is expandable by adding more IPUs to the set. The current system utilizes 5 IPUs which reside in 3 double-high VME card racks. It is estimated that 5 IPUs can perform over 30 million instructions per second (MIPs).

If size is not a limiting factor, 20-slot VME card racks may he used for one IPU. Up to four 68020 computer boards may be installed, giving the user approximately 12 MIPs of processing power per IPU. Sixteen slots remain available for memory, interface boards, or any other VMEbus module.

SOFTWARE

A Host system utilizes a menu-driven Test Plan Compiler which allows an operator to select the processes to be performed, define parameters, and input any special algorithms.

The compiler combines all of the real-time-run modules, assembles and links them, and then distributes the code into the IPUs over RS-232 links. The real-time modules are written in “C” and 68020 Assembly code.

EU tables may be down-loaded directly into an IPU’s memory. The size and number of tables allowed depend upon the raw sample size (bits/sample) and the amount of DRAM available. Table I shows the number of tables which may be stored in 2 megabytes of DRAM within each IPU. Additional memory modules may be added without increasing software.

TABLE I

Sample size	Table size	Number of Tables
8 bits/sample	256 bytes	8192
10 bits/sample	1024 bytes	2048
12 bits/sample	4096 bytes	512
16 bits/sample	65536 bytes	32
ANY COMBINATION ABOVE FILLING 2 MEGABYTES OF DRAM		

EU conversion equations may also be down-loaded into an IPU. However, table look-ups should be performed whenever possible since it requires less than 2 micro-seconds (μs) versus 33 μs for the calculation of a 5th order polynomial.

BENCHMARKS

Execution times of some processes are given in Table II. The times include all memory accesses as well as transfer times across two 48-bit-wide interface boards (16-bit tag, 3-bit data). The execution times are for a single Motorola MVME-130DOF 16.67 MHz computer module. Source code for the benchmarks is given in Appendix A.

Since the 63020 and 61831 operate in parallel, algorithms may be performed simultaneously. For example, the 68020 can perform an EU, WP, SUM, MIN/MAX, and check Phase Alignment while the 68931 is performing a Floating Point operation. In other words, derived algorithms are performed during the same time the cyclic and statistical processes are done.

TABLE II

68020 processes

Algorithm performed	Exec. times
EU (Table look-up)	3.8 μs
Wild Point	4.8 μs
Integer Summation	4.0 μs
EU, WP, SUM, MIN/MAX, Phase	12.0 μs

68831 processes

Floating Point Add	13.6 μs
Floating Point Mult	14.9 μs
5th Order Poly	64.9 μs

EXPANDABILITY

A telemetry preprocessor system which utilizes VMEbus-based 68020 systems is easy to expand and upgrade. Memory modules may he added at any time without additional software development. Other VMEbus modules like Digital-to-Analog boards and Discrete I/O boards may be installed.

The future outlook is very promising. 68020 microprocessors operating at 20 MHz and beyond will be available along with denser memory boards. As these boards are available, the user could upgrade the system and allow for more processing to be performed at a much higher throughput.

CONCLUSION

VMEbus-based systems using off-the-shelf computer modules can handle the real-time processing needs of modern telemetry systems. One hundred percent processing of data may be obtained during real time while maintaining throughput rates of 2 megasamples per second. These systems are low-risk due to the use of off-the-shelf hardware and available software development tools. They are easy to expand, and the growth potential appears to be unlimited.

ACKNOWLEDGEMENTS

The author wishes to thank Mr. Jay Lory of the Physical Science Laboratory for furnishing the Benchmarks given in Table II.

APPENDIX A

```
**** E.U. CONVERSION ****

LOOP:   MOVE.L      (A0),D0                   ; get raw data & FIFO status
        BMI.B       LOOP                      ; branch if FIFO is empty
        MOVEA.L     (A1),A2                   ; get packet address
        MOVE.L      ([4,A2],D0.L*4),([A2])    ; EU look-up & transfer
        BRA.B       LOOP                      ; next

**** WILDPOINT TEST ****

LOOP:   MOVE.L      (A0),D0                   ; get raw data & FIFO status
        BMI.B       LOOP                      ; branch if FIFO is empty
        MOVEA.L     (Al),A2                   ; get packet address
        MOVE.L      D0,D1                     ; copy counts
        SUB.W       4(A2),D1                  ; get slew rate
        BPL.B       WP_1                      ; branch if (+)
        NEG.W       D1                        ; make slew rate (+)
WP_1:   CMP.W       6(A2),D1                  ; test if slew rate above max allowed
        BGT.W       WILD POINT                ; branch out if wildpoint
        MOVE.L      D0,([A2])                 ; send raw counts out
        BRA.B       LOOP                      ; next

**** AVERAGE OVER n ****

LOOP:   MOVE.L      (A0),D0                   ; get raw data & FIFO status
        BMI.B       LOOP                      ; branch if FIFO is empty
        MOVEA.L     (A1),A2                   ; get packet address
        ADD.L       D0,4(A2)                  ; sum in counts
        SUBQ.L      #1,8(A2)                  ; end of refresh period ?
        BNE.B       LOOP                      ; no, next
        MOVE.L      4(A2),D0                  ; get sum
        DIVU.L      12(A2),D0                 ; get average
        MOVE.L      D0,([A2])                 ; send out average
        CLR.L       4(A2)                     ; zero sum
        MOVE.L      12(A2),8(A2)        ; restore n
        BRA.B       LOOP                ; next
```

**** SINGLE PRECISION FLOATING-POINT ADD ****
(operands are in memory)

```
        FMOVE.S     (A0),FP0            ; get operand
        FADD.S      4(A0),FP0           ; add
        FMOVE.S     FP0,8(A0)           ; store result
```

**** SINGLE PRECISION FLOATING-POINT MULTIPLY ****
(operands are in memory)

```
        FMOVE.S     (A0),FP0            ; get operand
        FMUL.S      4(A0),FP0           ; multiply
        FMOVE.S     FP0,8(A0)           ; store result
```

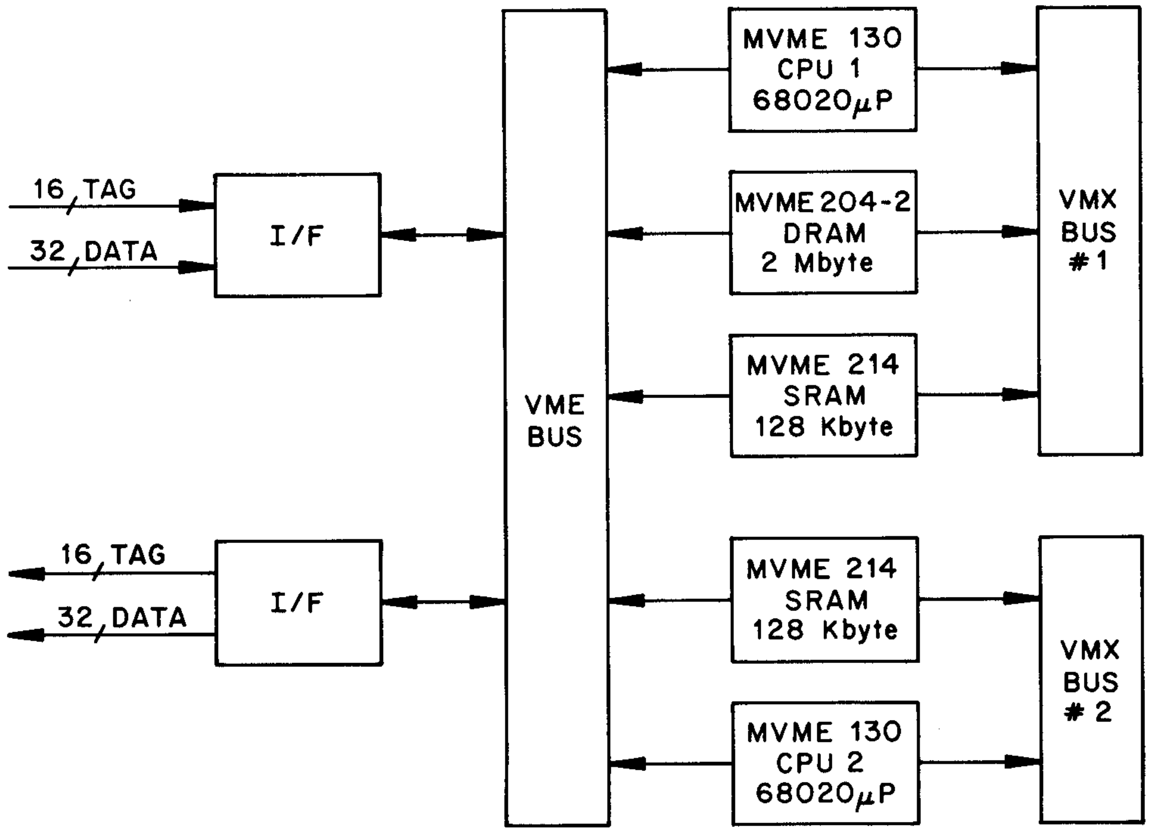

Figure 1. Dual Processor IPU Configuration