SLAC – PUB – 3680
May 1985
(E/I)

THE 3081/E PROCESSOR AND ITS ON-LINE USE*

P. RANKIN, B. BRICAUD, M. GRAVINA, P.F.KUNZ, G.OXOBY, AND Q.TRANG

Stanford Linear Accelerator Center
Stanford University, Stanford, California, 94305

and

P. M. FERRAN, A. FUCCI, R. HINTON, D. JACOBS,
B. MARTIN, H. MASUCH, AND K.M. STORR

CERN, CH-1211 Geneva 23, Switzerland

ABSTRACT

The 3081/E is a second generation emulator of a mainframe IBM. One of it's applications will be to form part of the data acquisition system of the upgraded Mark II detector for data taking at the SLAC linear collider. Since the processor does not have direct connections to I/O devices a FASTBUS interface will be provided to allow communication with both SLAC Scanner Processors (which are responsible for the accumulation of data at a crate level) and the experiment's VAX 8600 mainframe. The 3081/E's will supply a significant amount of on-line computing power to the experiment (a single 3081/E is equivalent to 4-5 VAX 11/780's). A major advantage of the 3081/E is that program development can be done on an IBM mainframe (such as the one used for off-line analysis) which gives the programmer access to a full range of debugging tools. The processor's performance can be continually monitored by comparison of the results obtained using it to those given when the same program is run on an IBM computer.

1. INTRODUCTION

The 3081/E is a second generation emulator of a mainframe IBM. One of the applications of such a processor is to increase the on-line computing power available to experiments and 3081/E's will form part of the data acquisition system of the upgraded Mark II detector for data taking at the SLAC linear collider. Since the introduction of the 168/E[1], much experience has been gained in making efficient use of such processors, and their value has been well established over a wide range of applications[2] including off-line event reconstruction , Monte Carlo studies, on-line triggering ,and filtering. This type of processor, unlike computers or commercial microprocessors, does not run an operating system nor is it directly connected to I/O devices. For the Mark II application a dual-ported FASTBUS interface will be provided to allow communication with SLAC Scanner Processors[3](SSP's) which accumulate data at the crate level and the experiment's VAX 8600 mainframe. The 3081/E will appear to be a slave device.

This paper will describe the 3081/E, concentrating on those features relevant to its use in high energy physics applications. The interfacing of the processor to a FASTBUS system and its use on-line will also be discussed.

* Work supported by the Department of Energy, contract DE-AC03-76SF00515.

2. THE PROCESSOR

The 3081/E project was formed to build on the experience gained with 168/E's and to produce a much improved IBM mainframe emulator. The style of simple,flexible interfacing to the host system has been retained. The new processor, however, has much more memory space, incorporates more IBM instructions, and has full double precision floating point arithmetic. Execution times have been substantially reduced. The time scale of the project was chosen to allow the use of recent electronic developments while making the processor available for use by LEP and SLC experiments. The design is very conservative and uses only off-the-shelf, multiple sourced TTL components, however, the processor has a modular architecture to make it easily upgradable.

The details of the processor and its architecture can be found in reference four so only a brief summary will be given here. There are two 64 bit wide operand buses, the ABUS and the BBUS(with associated 8 bit wide odd parity buses), which are connected to four execution units (integer,divide,multiply and floating point add/subtract), and also to the control and register unit, data memory, program memory, and an interface. The control and register unit serves four functions: it contains the microprogram address counter, conditional branching logic, the data memory address logic, and the register files. A microinstruction can transfer two operands simultaneously on the ABUS and BBUS from data memory and/or registers to an execution unit. The results from an execution unit are transferred on the BBUS to a register, to memory, or along with a new operand on the ABUS to another execution unit. Instructions are fetched on a third, 32 bit wide bus, the PMD bus.

The choice of a modular architecture helped tremendously in reaching the design aims of simplicity,reliability and ease of debugging. The first prototype was ready to run real programs in three weeks. FORTRAN simulations of each execution unit have made a valuable contribution to the designing and debugging. For example, only one design error was found in the Add/Subtraction execution unit when it was debugged, although this unit contains over 200 MSI circuits. This error was due to a single signal which had the opposite polarity in the hardware due to a mistake in the simulation. The divide board, which was the most difficult board to debug, initially took a week to get running. The second divide board only took a day to debug. In fact, it has taken far longer to develop the diagnostic software than to use it to successfully test the hardware.

Presented at the Conference on Real Time Computer Applications
in Nuclear and Particle Physics, Chicago, Illinois, May 20–24, 1985

2.1 MEMORY

Memory is one of the most important aspects of any computer or processor. In the high energy physics field, both the size of analysis programs and the quantity of data per event have grown so that the memory space needed is measured in units of Megabytes.

The memory of the $3081/E$ consists of static memory circuits which were chosen for their speed. Today they have 55 nsec maximum access and cycle time, come in packages of 16 Kbits, and cost about US$ 5,000 per Megabyte. The speed is important due to the fact that the performance of a processor tends to be dominated by its memory access time. This is because even with the best of compilers, a processor still obtains one operand (of the two used for an arithmetic instruction) from memory over 75% of the time. The fast memory and 64 bit data path to it is also the best solution for on-line applications which must support FASTBUS I/O rates.

A $3081/E$ memory board at present contains one half Megabyte of either program or data memory with byte parity. The processor can accept a maximum of fourteen memory boards or 7 Megabytes. 64K static memory circuits were introduced in 1984 and so will become reasonably priced during this year. Their use will lower the cost of the processor's memory and make it possible to have a processor with a 28 Megabyte storage capacity.

2.2 EXECUTION UNITS

For high energy physics code, good floating point performance is essential, especially because of the heavy use of trigonometric functions in most analysis codes for solenoidal detectors. Attempts to use commercially available microprocessors with their floating point co-processors have led to disappointingly poor performances.

The following sections give a short description of each of the $3081/E$ execution units. It will be possible to upgrade any of these units, if technological advances warrant the change, in order to achieve a better performance and/or lower costs.

Floating point add/subtract

A REAL*4 or REAL*8 add/subtract is done in 360 nsec (three clock cycles), including the reading of one operand from memory. The floating point compare instruction is one cycle shorter because it is necessary only to generate the condition codes and not a result. In addition, this execution unit can do an integer to floating point conversion in 360 nsecs.

Multiply

The multiply execution unit was designed to optimize the execution time of single precision instructions. 16 by 16 multiplier circuits are used so that INTEGER*4 and REAL*4 multiplies take 360 nsec including reading one operand from memory. Double precision multiplication is performed using an iterative technique which is reasonably fast (the result is available after only 4 internal cycles for an overall time of 720 nsecs including the reading of one operand from memory).

Divide

The divide execution unit does division iteratively, 2 bits per cycle which leads to a INTEGER*4 divide in 2.28 μsecs , a REAL*4 divide in 1.68 μsecs , and a REAL*8 divide in 3.6 μsecs. As will be seen later, pipelining of the instructions allows operations not dependant on the result of the divide to proceed during this time.

Integer

All integer instructions except multiplication and division are handled by the integer execution unit including the instructions with one–byte operands (LOGICAL*1 and CHARACTER*n) which are especially important for implementation of the instructions required by the FORTRAN 77 compilers. Both single word (32 bit) and double word (64 bit) shifts by any number of places are done in one cycle. Shift instructions are important for many on-line applications, when information often needs to be packed or expanded .

Optional units

Since there are PROM'S on each board to decode the microinstructions, it is possible to add other execution units to the $3081/E$ buses. These may duplicate existing units for debugging purposes or to increase the hardware available to do divides (for example). In addition, boards such as a matrix multiplier/accumulator for lattice gauge calculations may be specifically designed to match an application. For the moment special units are beyond the scope of the $3081/E$ project. They may also be unnecessary as the processor is inherently very fast.

2.3 THE MICROCODE, THE TRANSLATOR, AND INSTRUCTION PIPELINING

The processor's instruction set is not that of the IBM, but is its own microcode, which resembles that of a Reduced Instruction Set Computer (RISC)[5]. One could in principal write a compiler to generate the microcode, as was done with IBM's 801 project[6], instead it is generated by a program, called the Translator. This program reads IBM object code modules, translates them to object microcode, and links them together to form an absolute load module for the processor, thus using the IBM object code as an intermediate language. The source of the IBM object code could be the output of a FORTRAN compiler from any IBM compatible vendor or that of a linkage editor on either the VM/CMS, MVS, or MVT operating systems. For all practical purposes the translator step has little impact on the user. It can be looked on as a modified compile or link step. The user will be no more concerned with the $3081/E$ microcode than he would be about the object code from the compiler and he need not recompile his code before translating it.

In addition to converting the object code into $3081/E$ microcode the translator may also optimize the order of the instructions, pipelining them to yield faster execution times. This instruction pipelining is possible since each of the execution units is capable of operating on its operands internally so microinstructions may affect more than one board allowing parallel operations. There are several types of pipelining. Firstly there is pipelining of memory address calculations on the Control and Register board. Secondly, the Add/Subtract and Multiply execution units are capable of pipelining internally. That is, they can accept a new operand pair every cycle, then output the results in the next two cycles. Thirdly, one cycle can send an operand pair to say the add/subtract unit, and the next cycle can send an operand pair to the multiply. Fourthly, in the same cycle an execution unit can output results and another execution unit, or memory, can accept the results, thus overlapping input and output cycles. In addition, the separation of program and data memory and the separate program data bus means that program and data memories are accessed simultaneously.

Typically, a string of up to 100 or more IBM instructions may be microcoded and pipelined by the Translator. A string is defined as all the instructions between branch-in or branch-out points. In pipelining a string, the Translator may optionally use a number of the following techniques:-

1) Allocating extra floating-point registers. The 3081/E has 8 extra registers each of which may be allocated either for floating-point or integer usage.

2) Allocating extra integer registers. For example if the same (base+index) combination is used twice or more, it pays to add the base + index one time only and keep the result in a temporary register.

3) Optimal sequencing of Divides. Divides require many cycles and only one divide may be performed at a time. In the case of two or more independent divides, the Translator sequences them in such a manner as to produce the shortest overall pipelined string.

4) Critical Path Analysis of strings. The Translator analyzes which IBM instructions could logically be performed concurrently, and may optionally reorder the IBM instructions in a string, so that those instructions on the 'critical path' get selected first for being moved as high as possible during the pipelining operation.

5) Combining certain IBM instruction pairs into a single 3-operand pseudo instruction, before pipelining. for example:-

LE REG,MEM2
DER REG,SREG

may be changed to:-

DERX REG,MEM2,SREG

The effect of this is that although the result of the divide must be stored in **REG** the dividend does not need to be stored there initially. This means that any operations involving the prior contents of **REG** do not need to be completed before the divide is started but may be done in any cycle preceding the end of the divide.

6) Optional deletion of NOP instructions, ie branches on condition zero.

7) Recognition of special sequences. For example the 3081/E has a hardware fix-to-float capability, which can replace the several instructions used on the IBM machine.

The microcode does require more memory space than the original object code. However, the expansion factor depends on how well the code pipelines, and is typically under two and never more than three.

2.4 PERFORMANCE

It is difficult to predict the execution speeds of code on the 3081/E in advance of actually running it. This is because not only do different instructions take different numbers of cycles to complete, leading to a dependence on instruction mix(in common with many other processors), but also the extent to which the code can be pipelined varies (IF statements break the pipelining for example). If no pipelining occurs a lower bound can be put on the processors performance; in this case it is equivalent in processing power to an IBM 370/168. If, alternatively, it is assumed that pipelining occurs to such an extent that every instruction takes effectively 1 cycle, then this implies execution times 2.5 times faster than an IBM 370/168; an upper limit on processor performance that can not be realistically expected.

In addition to these theoretical estimates we have also run some tests directly comparing the processor to an IBM 3081KX. We require that the emulator also gets exactly the same results as the mainframe. Performance factors of about 50% of the IBM 3081KX are typical (for example , for a calculation involving finding the sum of squares of sines and cosines), and have reached 65% (for a large numerical analysis program). It is worth emphasizing that these numbers may well be improved upon . The translator program is still under development and the microcode can be expanded to include more of the pipelined instruction combinations discussed above.

One can conclude, therefore, that the performance of the 3081/E is at least that of an IBM 370/168 (or about four times that of the VAX 11/780). For typical high energy physics event reconstruction code, in which most of the execution time is spent in floating point loops, it is frequently 50% faster .

3. ON-LINE USE

A multiple 3081/E system is being planned to form part of the Mark II upgrade's data acquisition system. Initial plans call for the introduction of two processors into the system by Fall 1985, when the detector is to be checked out at PEP. The need for further processors for SLC use will be determined by their performance during this PEP run. Much expertise in using such multiple processor systems has been gained with the 168/E[8] and the 3081/E system will build on this by using the same overall philosophy with only the few changes which experience has shown to be necessary. The bus interface to the processor will be FASTBUS.

The 3081/E's represent a significant amount of the on-line computing power available to the Mark II upgrade. They will be used to reduce processing intensive loads on the experiment's VAX 8600 computer. One function that the 3081/E's can perform is to combine the two sets of data coming from the Drift chamber[7] to form one output record. Drift time measurements will be made using four FASTBUS crates of TDC's the TDC modules will be readout by SLAC scanner processors which will process the data and order it. Information on the energy deposited by tracks in the drift chamber will be obtained using 18 FASTBUS crates of dE/dX modules. These modules will also be readout using SSP's. The SSP's will also do some processing of the dE/dX data, but, in order to extract as much information as possible from it, the more complex pulses will be processed by the 3081/E taking full advantage of its large memory and its floating point instruction set. The aim will be to find the areas of pulses, extract timing information, and to flag possible double hits which cannot be separated by the TDC system (and if possible resolve them), before the data is transferred to the VAX.

In addition, the 3081/E's can take over from the VAX 8600 the heavy processing load associated with on-line tracking. An important characteristic which particularly suits them for use in tracking is their emulation of the IBM instruction set. The off-line tracking programs run on an IBM which means programs can quickly be moved from the off-line to the on-line environment. Once the program development has be done on the IBM 3081KX, making full use of the debugging tools and diagnostics available, the program can easily be converted for on-line use.

Consider an off-line tracking program . The input data is read off a tape, processed, and the results output to tape or

disk. Most of the processing time is involved in finding hits and doing the associated tracking This stage may well not involve I/O unless it is necessary to print warning or error messages. The tracking code can easily be isolated by forming it into a stand-alone subroutine. A skeleton main program is needed to do the event read-in, call the tracking subroutine, and output the final results. In the on-line environment the data is read-in directly from the data acquisition system. In the Mark II case the raw data will be collated in system scanners and transferred into the memory of a $3081/E$ processor. In order to do the tracking in a $3081/E$ the following steps are necessary.

1. The subroutine which does the tracking must be set up so that any information it uses is transferred to it via common blocks. The warning and error messages should be modified to store an error code and any associated information in a common block, since all print statements must be deleted. The results of the subroutine must also be put into common blocks. These modifications are usually made with little difficulty by anyone with some knowledge of the program, the requirement for data to be input and output via common blocks, for example, is usually already met.

2. The subroutine must be translated on an IBM computer. This converts the program into $3081/E$ microcode and sets up a set of "local" constants used by the program for downloading into the data memory of the $3081/E$. In addition the translator fixes the location of common blocks and provides a header containing this information which can be used by the host computer to load and read these common blocks. (This is why information used by the host computer needs to be in common blocks, otherwise minor program changes would change the address of these variables requiring programs which used them to be edited as well). The translator output must then be transferred to the host computer.

3. Before data taking begins the program and data memory of the $3081/E$ must be loaded with the translated files. This can be done once/shift, whenever a new run is initialized, or whenever the program is changed. The program and data memory are loaded in the same way except for the setting of an invert bit to determine which is loaded. During data taking the program memory cannot be over-written (unless the invert bit is set by error), and the interface is designed so that an attempt to write data into a protected region of $3081/E$ data space will result in a FASTBUS error response being sent to the master device.

4. The host computer supervises the transfer of raw data into the processor. In the Mark II case, a FASTBUS master, the SSP must determine which processor, if any, is available to receive data. The $3081/E$ will then be loaded and a command issued to start program execution.

5. When execution is completed the results are transferred out of the $3081/E$ into the host computer. They may then be outputed onto tape or used by the host computer in monitoring the performance of the detector. The interface used by the Mark II system is designed so that the FASTBUS master device has access via the interface to status registers and can check for normal program completion. The user has the option, for example, of either stopping program execution on errors such as division by zero or of latching such errors and checking for their occurrence following program completion.

Continual monitoring of the processors performance can be made by directly comparing results running the same program on the $3081/E$ and the mainframe.

4. INTERFACE

Since the processor does not run an operating system or have direct connections to I/O devices an interface is needed. For Mark II use a FASTBUS interface will be provided to allow communication with SLAC Scanner Processors(which are responsible for the accumulation of data at a crate level) and the experiment's VAX 8600 mainframe. This interface will be dual-ported allowing each $3081/E$ to be a slave on two FASTBUS cable segments. The two ports are symmetrical, neither having higher priority.

The Mark II interface to the $3081/E$ processor is of the same style as $168/E$ interfaces[9]. That is, either the CPU or the interface has control of the internal buses. Any $3081/E$ interface can be thought of as comprising of two parts. The first of these takes signals from the external bus, eg FASTBUS, and is application specific. This part of the interface converts external signals into a predefined form for input into a second board. This board communicates with the internal buses of the $3081/E$. One of the design aims of the FASTBUS interface group has been that this board should be capable of forming part of any $3081/E$ interface. During hardware debugging when it may be useful to interface to an IBM PC rather than the FASTBUS SYSTEM only the first board of the interface needs to be changed.

The interface can be used both in an on-line and in an off-line multi-processor environment. The dual-porting allows for the simultaneous loading of one processor and unloading of another. The loading and unloading will not necessarily be done by the same FASTBUS master. A FASTBUS master wishing to attach to the processor, either to read it or to load it, therefore needs to know not only if the processor is executing a program but also if another master has attached the $3081/E$. For this reason, one register (CSR0) is provided in the interface which is accessible from each cable segment regardless of the connection state and processor status.

When the processor is not running, all of the processor's memory is directly addressable through the interface. The design aims to minimize the time needed for FASTBUS block transfers. The transfer rate to or from the processor could be over 32 Megabytes per second if FASTBUS cable segments were sufficiently fast. The memory cannot be accessed when the processor is running. During data acquisition areas of data memory can be given a protected status by setting an allowed address range in interface control registers.

Some features are incorporated into the interface to make it easier to debug the processor and/or programs. The interface can halt the processor if there is a parity error on the ABUS, BBUS, or PMD bus. It also has registers to allow one to halt the processor when certain conditions arise in a way similar to the Program Event Recording (PER) registers of IBM mainframes. For example, there is a stop on a store within an address range, and a stop on modification of certain registers.

5. CONCLUSION

The 3081/E project was formed to prepare a much improved IBM mainframe emulator for the future. The time scale was chosen to allow it to use recent electronic developments but to still be available for LEP and SLC experiments. It has a modular architecture to make it easily upgradable. The design is based on a large amount of experience in using the 168/E processor to increase available CPU power in both on-line and off-line environments. The processor is at least equal to the execution speed of a 370/168 and up to 1.5 times faster for heavy floating point code. A single processor is thus at least four times more powerful than the VAX 11/780, and five processors on a system would equal at least the performance of the IBM 3081KX. With its large memory space and simple but flexible high speed interface, the 3081/E is well suited for the on-line and off-line needs of high energy physics in the future. Prototype wire-wrap processors are running at SLAC and CERN. The first multiwire boards are now being debugged, and the processors are becoming available for general use.

6. ACKNOWLEDGEMENTS

We would like to thank David Leith for his support, and Don McShurley and Richard Bacon for their work on the processor. We would also like to thank Loy Barker for his work on the FASTBUS interface and Andy Lankford and Leo Paffrath for their contributions to the design discussions. The 3081/E project is being carried out as a collaboration between SLAC and CERN DD division and the work has been divided equally between them.

REFERENCES

1. Paul F. Kunz, "The LASS hardware processor", Nucl. Instr. Meth. 135, 435 (1976).
2. P. F. Kunz, "Use of Emulating Processors in High Energy Physics", Phys. Scr. 23, 492 (1981).
3. H. Brafman *et al.*, "The SLAC Scanner Processor", *IEEE transactions in Nuclear Science, February, 1985.*
4. P. F. Kunz *et al.*, "The 3081/E Processor", *Proc. of the Three Day In-Depth Review on the Impact of Specialized Processors in Elementary Particle Physics, Padova, Italy, March 23-25, 1983.* P. F. Kunz *et al.*, "The 3081/E Processor", SLAC PUB-3332, April 1984. CERN DD/84/4, April 1984.
5. D. A. Patterson and C. H. Séquine, "RISC-1: A Reduced Instruction Set VLSI Computer", *Proc. Eighth Ann. Sym. on Computer Architecture, May, 1981.*
6. G. Radin, "The 801 Minicomputer", IBM J. Res. Develop. 27, 237 (1983).
7. G.Hanson et al. "A New Drift Chamber for the Mark II at the SLC", SLAC PUB 3317, April 1984.
8. J. T. Carroll, M. DeMoulin, A. Fucci, B. Martin, A. Norton, J. P. Porte and K. M. Storr, " Data Acquisition using the 168/E", *Proc. of the Three Day In-Depth Review on the Impact of Specialized Processors in Elementary Particle Physics, Padova, Italy, March 23-25, 1983.*
9. D. Bernstein, J. T. Carroll, V. H. Mitnick, L. Paffrath and D. B. Parker, "SNOOP module CAMAC Interface to the 168/E Microprocessor", IEEE Trans. Nucl. Sci. *NS-27*, 587 (1980).