1980

Mice, a fast user-microprogrammable emulator of the PDP-11

Blanc, Antonioz J.

CERN Libraries

http://hdl.handle.net/11728/6632

Downloaded from HEPHAESTUS Repository, Neapolis University institutional repository
MICE, A FAST USER-MICROPROGRAMMABLE EMULATOR OF THE PDP-11

J. Anthonioz-Blanc, C. Halatsis*, J. Joosten, M.F. Letheren, A. van Dam†, A. van Praag, C. Verkerk

This report is a preliminary version of a forthcoming paper describing MICE. In its present form it was made available at the Poster Sessions of the International Conference on Experimentation for LEP, Uppsala, 16-20 June 1980.

* permanent address: NRC Democritos, Athens
† permanent address: Brown University, Providence
MICE, A FAST USER-MICROPROGRAMMABLE EMULATOR OF THE PDP 11.

CERN, Geneva, Switzerland.

Abstract.

A fast processor is described which emulates a large part of the PDP 11 instruction set and which is also user-microprogrammable. It is intended for on-line filter applications, when decision times in excess of 100 microseconds are adequate. The machine runs PDP 11 code three times faster than a PDP 11/70 with 100% cache hits and can be equipped with interfaces to read-out systems with speeds of approximately 100 ns per data word.

Introduction.

A number of microprogrammable processors for real-time processing of high-energy physics data have been developed over the last few years [1-5]. The modern LSI components, such as bit-sliced units for arithmetic/logic and for program sequencing, make it rather easy to put together a digital processor in a relatively short time. The bipolar bit-slices are capable of running at speeds 5-10 times higher than the single chip microprocessors, which makes their use very attractive for fast pre-processing of experimental data in those cases where programmability is a necessity.

A machine built from bit-slices and other modern components will however only be microprogrammable normally. Programming can then only be done at a very low level, requiring detailed knowledge of the internal workings of the ICs, of the way they are interconnected and of the timing relations to be observed. The designer of the device will himself not necessarily encounter great difficulty in microprogramming his machine, but another user may and generally will find it complicated and tedious. In addition, the programming and debugging tools at the disposal of the user are often very primitive.

The processors which emulate large scientific computers - 168E [6] and PUMA [7] - have solved the programmability problem, but these solutions have concentrated on number crunching applications and neglected the on-line use with its interfacing problems.

We felt that also in on-line applications profit should be taken from the advantages of emulation. An earlier emulation of the PDP 11 [8] had not attained high speed, which was compensated in this case by using several units in a multiprocessor configuration. MICE (MICroprogrammable Engine) was therefore developed with three main objectives in mind:

* permanent address: MRC Democritos, Aghia Paraskevi, Athens.
+ permanent address: Brown University, Providence, Rhode Island.
- make programming of the device easier by emulating a well known and widely used minicomputer, the PDP 11. This ensures that the machine can be programmed in any language for which a compiler, interpreter or assembler exists. Program development can be done on a host computer of the PDP 11 family, equipped with all peripherals and back-up storage convenient for the purpose. Only running object code of application programs need then to be transported.

- attain execution speeds far in excess of those of the fastest member of the PDP 11 family. Whereas high speed is essential for real-time filter applications, the algorithms are sufficiently simple and small so that they do not need large memory and arithmetic resources. For the sake of simplicity and speed the machine can therefore be limited to a small configuration (28 K memory) with fixed point arithmetic only.

- provide interfaces to fast read-out systems, so that the device may be effectively and efficiently used in on-line filtering and event selection.

These three main objectives did not exclude that the user may have recourse to microcode if his application is extremely time-critical. In most cases he may limit the tedious task of writing microcode to certain, generally small, parts of his program, where most of the execution time is spent: inner loops or frequently called subroutines. If, as is often the case, more than 90% of the execution time is spent in less than 10% of the code, the amount of user-written microcode can be limited and precious programmer or physicist time saved.

These main objectives led to a number of design decisions which have been described previously [9]. The present paper will not enter into details of the design, but rather give a user-oriented description, together with some performance figures.

Hardware

MICE has been built with Motorola's ECL 10800 bit-slice family and with fast ECL 10146 memory chips. As described in [9] the CPU has been designed for efficient execution of PDP 11 code, which necessitated the implementation of additional logic and data paths, not foreseen in the 10800 family. A block diagram is shown in figure 1, a more detailed diagram of the CPU is given in figure 2. The microcycle time is 105 ns.

The machine has two separate memories:

- the target memory, 16 bit wide, holds the PDP 11 code (both program and data). One board contains 4 Kwords of fast (40 ns) ECL memory, but boards containing 16 Kwords of slower MOS memory can also be used. The total size of the memory is restricted to 32 K of which 28 K can be effectively used.

- the writeable control store, 120 bits wide, 1 Kword long. The microstore contains the microprograms. The emulator, e.g. the microcode needed to interpret and execute PDP 11 instructions occupies 400 words. 50 words are reserved for a diagnostic routine. 512 words are available for user-written microcode. The WCS can be extended to 4 Kwords if required.
The two main internal buses of the CPU are extended (XIB and XOB, figures 1,2) and can be used for the attachment of special purpose processing elements. A fast multiply unit (210 ns multiply time for two 16 bit numbers) is at present under construction. Units performing more complicated algorithms can also be envisaged.

A simplified version of the Unibus is also provided: standard PDP 11 peripherals can be attached to it, provided they do not require DMA transfers.

Direct Memory Access to the memory can however be made via special interfaces and two of these are foreseen at present. One is an interface to the RMH read-out system [10], which will allow this system to run at about 100-120 ns per valid data word. An interface to a Romulus/Remus branch [11] is being constructed.

MICE is controlled from a host computer, via Camac. An interface is provided which is connected to the Camac Dataway. This interface provides the facilities normally found on an operator console: load program counter, halt, start, continue. The Camac connection is also used to load programs into the two memories, or to dump their contents. This interface also contains the usual facilities to load, read and test registers for handling of interrupts from and to the processor. For debugging purposes the Camac interface can suspend the clock, or run it for n cycles.

MICE has been implemented on large boards (318x265 mm2). Wirewrap technique has been used throughout. When sufficient care is taken in the layout of ICs and interconnecting wires, the wirewrap technique proves to be perfectly adequate for wiring of large dense boards of ECL logic. The CPU together with the microstore occupies three boards, the different interfaces one each. A configuration with a fast multiplier, a Romulus interface and 4 K fast memory thus occupies 7 boards. The cost of such a configuration is approximately 13 K$, including power supplies and crate.

Emulation of the PDP 11

The PDP 11 instructions are decoded by hardware into a starting address of a micro-routine which then performs the operations needed for its execution. The micro-instruction word is very wide (120 bits are used) and divided into a number of independent fields. This makes it possible to perform many elementary operations simultaneously, which improves speed.

This concurrency at the microcode level is efficiently used to pipeline the execution of target (PDP 11) instructions. The execution of an instruction normally requires four phases: instruction fetch, instruction decoding, execution proper and update of the program address. In MICE these phases are handled simultaneously, for consecutive instructions. Figure 3 shows this pipeline scheme: during microcycle i instruction j+2 is fetched, instruction j+1 is decoded and the corresponding micro-instruction is fetched; the operation asked for in instruction j is performed and the program counter is prepared for fetching of instruction j+3. Four target instructions are therefore normally in the pipeline. Figure 4 illustrates a typical microprogram sequence for an ADD
instruction, involving fetching an operand from and writing a result to memory. The pipeline scheme is the main contributing factor to MICE's speed.

Most fixed point PDP 11 instructions are emulated and MICE therefore behaves like a PDP 11. There are however a few minor differences:
- byte instructions are not implemented (except MOVB and SWAB)
- memory management has not been implemented, maximum memory size is thus 32 K.
- there is only one interrupt level.
- the simplified Unibus has no bus mastership arbitration. As a consequence, this bus does not support DMA transfers.

None of these restrictions are important for the applications for which MICE is intended.

On the other hand, to improve the performance of MICE in on-line filter applications new instructions can be introduced. For example, we implemented two new opcodes:

- Repeat instruction (RPT n). When encountered in a program, this causes MICE to execute the next instruction n times. This feature is useful to transform short loops into a single instruction, thereby reducing overhead. The following sequence:

  MOV N,R2
  LOOP ADD R0,(R1)+
  SOB R2,LOOP

  can be rewritten as:

  RPT N
  ADD R0,(R1)+

  Whereas the first sequence takes 6N+2 microcycles, the second only takes 3N+3.

- Jump to Microcode (JMC Label). When this instruction is encountered, MICE will start to execute (user-written) microcode. This new instruction thus allows the execution of mixed code. Tight inner loops or time critical parts of an algorithm can be speeded up. The speed-up factor depends on the program and cannot be predicted with precision. A factor 2 seems reasonable but 5 might be possible under very favourable circumstances. This is for instance the case when an array in memory must be scanned for an element smaller than the value in R1. The loop can be coded in a single micro-instruction; the execution is then 6 times faster than for the assembly code.

Support Software.

---------------

The well-known advantage of emulating an existing machine is that the support software needed for program development does exist. A large machine can be used for file storage, program editing, assembly or compilation as well as for testing and debugging. Programs can be written in any language for which a compiler exists.

Software development is done on the PDP 11 control computer, or alternatively on a PDP 11 in the computer centre, connected to the control computer via CERNET.
For writing microcode we have adapted a commercially available cross-assembler [12]. This assembler has extensive macro facilities which allowed us to define a language for writing microcode which is relatively convenient for the user. The actions do not need to be described at the lowest level (e.g., control of single gates), but mnemonics describe actions which are meaningful to a programmer. The language does however not hide the basic architecture of the machine and thus by combining several mnemonics and parameters into a single micro-instruction full profit can still be taken of the inherent parallelism of the machine. Extensive error checking in the assembler protects the programmer against using conflicting operations and other errors.

During the design of MICE we have made extensive use of a general purpose architecture simulation facility: ISPS [13]. Microcode had been simulated and debugged before any hardware was available. This tool is also expected to be useful for debugging user-written microcode.

As mentioned before, MICE is controlled and loaded from Camac. A PL-11 task running under RSX11S on a 28K PDP 11 handles this. At present this task is being rewritten in a transportable language, using standard ESONE Camac calls. With the help of a microcoded diagnostic routine in MICE this task can trace at each microcycle the contents of all internal registers in MICE.

Performance

MICE runs at its design speed of 105 ns per microcycle. This results in a speed three times that of a PDP 11/70 with 100% cache hits. Comparisons with a PDP 11/40 have shown a speed factor of 12. This figure is in agreement with the number above.

During tests MICE has run programs written in PAL 11, in PL-11 and in Fortran. A small run-time Fortran system has been written, which is resident in MICE (size 256 words). The simplified Unibus connection has been used to drive a CC-11 Camac interface.

Conclusion

MICE has shown that very fast emulators can be built, provided that some non-critical restrictions are acceptable for the envisaged applications. Emulation of a widely used minicomputer is attractive for on-line filtering of events, when decision times in excess of 100 microseconds are acceptable and when ease of programming is of prime importance. For time-critical code (often a small fraction of the total program, e.g., inner loops), the performance can be enhanced by microcoding or the construction of special functional units. Thus the MICE design is a good compromise between the flexibility of minicomputer software and the speed of hardwired logic. For decision times well below 100 microseconds the only alternative is hardwired processors, which lack of course the flexibility of a programmable device.
References

1. T. Lingjaerde, A fast microprogrammable processor, internal report CERN DD/75/17.

Figure Captions.

Fig. 1. MICE configuration. Data from the experiment are stored in TM (target memory) via a fast DMA interface.
Fig. 2. Block diagram of the CPU.
Fig. 3. Target instruction pipeline scheme. The target memory is busy for every cycle (each column contains either a target fetch (F) or a write (W) operation).
Fig. 4. Typical program sequence and its breakdown into microcycles.
NOTE: SPECIAL UNITS (E.G. MULTIPLIER) CAN BE CONNECTED TO THE EXTENDED INTERNAL BUSES OF THE CPU

FIG. 1 - MICE CONFIGURATION
FIG. 2 - MICE CPU
0 1 2 3 4 5 6 7 8 9 Time slots (105ns)

E0
Df1 E1
F2 Df2 E2
PF3 F3 Df3 E3
PF4 F4 Df4 E4 W4
PF5 F5 Df5 E5 W5
PF6 F6 Df6 E6 W6
PF7 F7 Df7
PF8 F8
PF9

Notes: Source and Destination operand fetching is not shown.

Ej : Execute target instruction j
Dfj : Decode instruction j and fetch micro-instruction
to execute target instruction j
Fj : Fetch target instruction j
PFj : Prepare fetch address of target instruction j
Wj : Write result of instruction j in target memory

FIGURE 5 - TARGET INSTRUCTION PIPELINE SCHEME
FIGURE 4 - TYPICAL MICROPROGRAM SEQUENCE