A microprocessor architecture for digital device implementation*

by THOMAS L. BOARDMAN, JR.

University of Colorado
Boulder, Colorado

ABSTRACT

A microprocessor based architecture for the implementation of digital logic devices is presented. The architecture facilitates replacement of portions of the hard-wired logic chains within the device with groups of microinstructions called kernels. Multiple modules of a device may share the processor simplifying interface problems such as buffering, interlocking, and sequencing. This greatly reduces overall package count, power consumption, system complexity, and therefore system debug time and cost while improving system flexibility with the programmable control store. A three dimensional graphics display processor for the Tektronix 4014 terminal, built using Intel 3000-series microprocessor elements, is discussed demonstrating the viability and benefits of this architecture. Such processor elements permit approximately 250,000 twenty microinstruction kernels to be executed in place of hard-wired logic per second.

INTRODUCTION

The control of basic digital logic functions such as sequencing, data flow, and arithmetic unit operation by bits in a programmable control store is at least as old as the EDSAC II (1953). Until recently, however, this microprogramming of logic functions served largely to improve system design flexibility, not to reduce size, power consumption, cost, or improve performance. The development of microprocessors using MSI and LSI technologies has provided these *additional benefits* and therefore significantly affected the digital logic design process.[1]

Initial utilization of these microprocessors,[2-4] centered around the Intel 8080 and similar hundred thousand instruction per second processors, was characterized by replacement of large sections of digital logic with serial execution of stored programs interfaced to the outside world and internal SSI logic. Such utilization made products such as calculators, point of sale terminals, adaptive traffic control systems, etc. economically feasible and will undoubtedly continue to represent a major part of the intelligent device market. They are, however, limited to applications requiring at most several thousand operations per second since each operation must be implemented as a series of machine instructions.

Recent bipolar LSI technology has produced processors such as the Intel 3000 and AM2900 series devices which are capable of executing several million instructions per second. These processors, while capable of handling orders of magnitude faster applications, are multi-package systems and therefore are not necessarily as cost or performance effective. In addition, since their multipurpose nature demands many internal logic levels between input and output, they can never completely duplicate the *performance* characteristics of SSI logic.

This paper describes an architecture for the interconnection of such bipolar microprocessors and SSI logic which can provide major logic function unit replacement for low speed-requirement operations, minor function replacement for higher speed-requirements, and direct logic execution of time-critical operations. Examples of these modes include matrix multiplication every several milliseconds (hundreds of instructions), multiple-bit shift or buffer push/pop every several microseconds (several instructions), and bit serialization for disk transfers (hardwired logic).

SYSTEM ARCHITECTURE

The microprocessor system referenced herein as an example of the architecture being presented was designed to support eight independent logic devices such as disk units, communication lines, CRT's, and multiply-divide units. It was implemented using Intel 3000 series microprocessor elements with a sixteen-bit word size (eight two-bit arithmetic unit slices) and a sixty-bit microinstruction word. Use of Intel components and these specific field sizes should not be considered characteristics or requirements of the basic architecture.

Whether it be along communication lines or wires etched in a single circuit board, signals must be bussed between the processor and the various devices of which it is a part. The seventy bus lines required by this implementation are shown in Figure 1. Thirteen lines are used for power

* This report was prepared as a result of work performed under NASA Contract No. NAS1-14101 while the author was in residence at ICASE, NASA Langley Research Center, Hampton, VA 23665

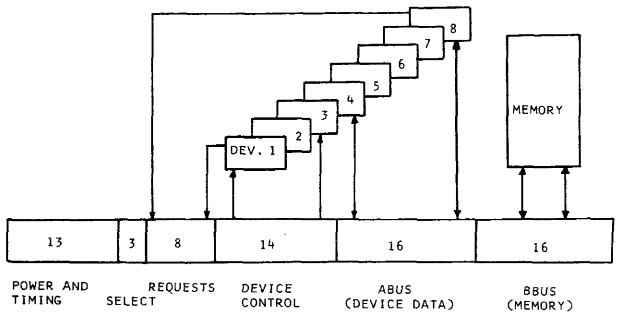


Figure 1—System bus structure

distribution and timing signals which are described below. Three lines specify which one of the eight devices is being selected by a particular instruction. The following eight lines allow the devices to request (interrupt) the processor for execution of specific groups of logic-replacement microinstructions. The interrupt structure, which makes the processor appear as a dedicated slave to each device, is also described below. The following fourteen bus lines are used to control the devices by initiating transfers to and from the processor and operations at the various points in the device logic chain. The next sixteen bits provide a bidirectional data bus for inputs to and outputs from the processor. Finally, the scratchpad read-write memory associated with the processor and thereby available to all devices is addressed and accessed over a second bidirectional bus.

As with most bipolar microprocessors, the arithmetic and processor (next address calculation) functions are performed by separate chips within the Intel 3000 family. Figure 2 shows the specific configuration of these chips. Figure 3 lists the various fields which make up the sixty-bit microinstruction. Many of these are specific to the 3000-series elements and will not be described in detail. Others will be referenced in the more detailed description of the processor implementation which follows.

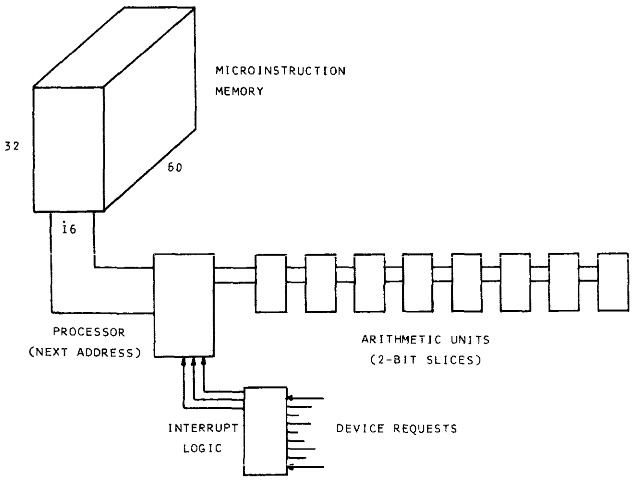
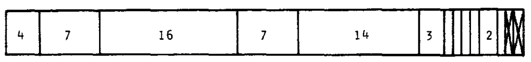


Figure 2—Microprocessor configuration



(4)	CARRY CONTROL	(1)	INTERRUPT DISABLE
(7)	NEXT ADDRESS FUNCTION	(1)	ARITHMETIC UNIT DISABLE
(16)	MASK INPUT TO PROCESSOR	(1)	BRANCH ON CARRY IN
(7)	ARITHMETIC UNIT FUNCTION	(1)	BRANCH ON CARRY OUT
(14)	DEVICE CONTROL	(2)	MEMORY CONTROL
(3)	DEVICE SELECT	(1)	ZERO PROCESSOR INPUTS

Figure 3—Microinstruction fields

PROCESSOR-DEVICE COMMUNICATION

Communication between the microprocessor and each of the devices, which occurs across the seventy-line bus described above, is controlled by the timing signals shown in Figure 4. The system is driven by a single twenty megahertz CLOCK to insure that the devices and processor remain synchronized.

An individual microinstruction begins execution with the leading edge of the CPU CLOCK. This initiates the next address calculation in the processor chip requiring approximately fifty nanoseconds. Once the address is calculated, it is presented to the (60-bit wide) microinstruction memory and the instruction is fetched. This requires sixty nanoseconds for the memory used in this implementation. At the same time, the bidirectional busses are switched to provide input from the devices and scratchpad memory to the arithmetic unit bit-slices. As the instruction is being fetched, an END CYCLE pulse is sent out terminating execution of the previous instruction. Although all devices receive this signal, it means only that they must relinquish the bus and access to the processor. Internal logic chains and interaction with their external devices may continue.

Once fetched, a portion of the microinstruction is LATCHED to permit overlapping of this instruction with the next instruction fetch. The remainder of the instruction is presented to the processor elements directly and to the devices over the bus as indicated in Figure 1. The new

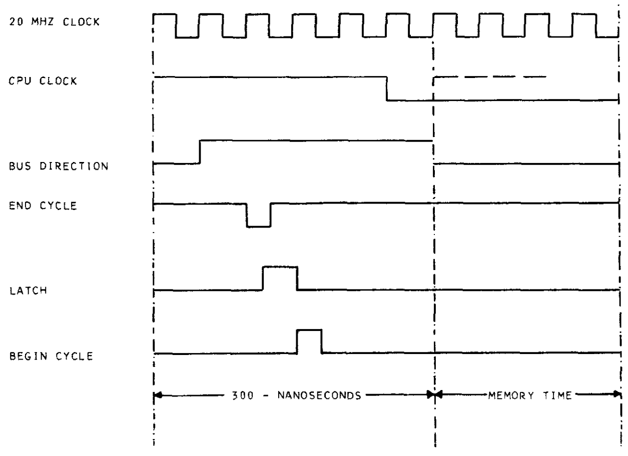


Figure 4—Microprocessor timing

instruction is then initiated with the BEGIN CYCLE pulse causing a specific device indicated in the instruction word to be selected. If requested by the appropriate control bits, the device immediately places data on the bus as input to the arithmetic units. After a settling time, the falling edge of the CPU CLOCK initiates execution of the microinstruction specified in the instruction word. As indicated, the arithmetic unit has access to both device data and memory data for that execution.

After approximately fifty nanoseconds, the results of the arithmetic unit execution are sent to the device selected and to memory as the BUS DIRECTION is reversed. If this is a memory reference instruction, the cycle is extended until the memory signals that it is complete. Otherwise, the CPU CLOCK rises immediately and the microinstruction execution process is repeated beginning with the next address calculation.

Notice that the device can specify input to the arithmetic unit, wait for it to be processed, read the result, and perhaps read a resulting value from memory (since the bidirectional busses again reverse) before the END CYCLE pulse terminates the instruction. This represents a significant characteristic of the architecture since most processors permit only read device, or write device, or access memory on a single cycle. In the class of interfacing problems for which this architecture was intended: read, process, and write operations are very common.

SYSTEM INTERRUPT STRUCTURE

The previous section described the event sequence for the execution of a single instruction. Since the next address opcode is part of each microinstruction (Figure 3), sequential execution of instructions is unnecessary. Groups of microinstructions required to perform a specific function can, however, be thought of as logically sequential kernels. These kernels perform operations for the digital devices to which the processor is connected ranging from single instruction shift operations to complex operations such as a matrix multiplication requiring scores of instructions.

To maximize the usefulness of the microprocessor to the individual devices, execution of these kernels must be possible at any point in a device's logic chain. The hardware interrupt structure provides this capability. In parallel with the execution of each instruction, the interrupt logic shown in Figure 5 monitors requests coming across the bus from all (eight) devices. At the point in the execution cycle immediately after the processor has computed the next address, the highest priority request is compared against the priority (number) of the device currently selected. If the requesting device is of higher priority, the computed next address is optionally stored in scratchpad memory and an instruction dedicated to handling interrupts from the requesting device is fetched.

The fetched interrupt handling instruction causes the contents of a scratchpad memory location (interrupt vector) dedicated to the interrupting device to be read, and branches to an instruction common to all device interrupt

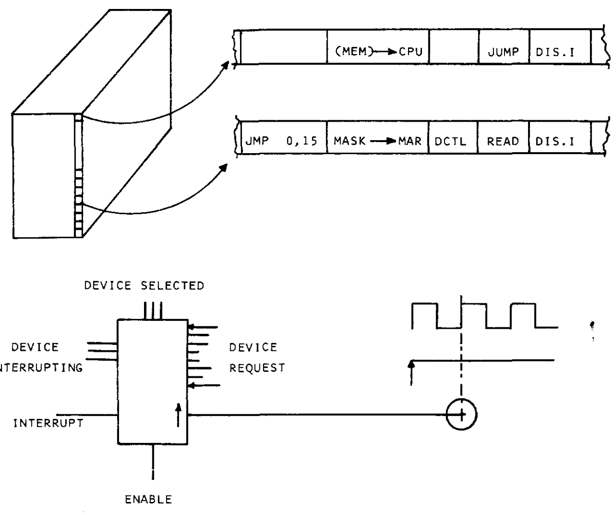


Figure 5—Interrupt structure

handling. This second instruction copies the value read, which is the microprogram address to be executed next for that device, into the processor so that it can be used as the next address executed, as depicted in Figure 5. Processing (kernel execution) for the interrupting device begins on the third instruction after the interrupt was received.

Any kernel can include instructions to change the contents of the interrupt vector and therefore point to a new kernel to be executed following the next interrupt from its device. This allows the processor to perform widely varied tasks for a device at different points in its logic chain. Since each microinstruction includes fourteen device control bits (Figure 3), the processor can provide not only logic simulation but also sequencing of the logic chains within the device.

Return from interrupts is handled by the same logic and procedure by which they are initiated. The end of each interrupt processing kernel is a branch to a NOP instruction of lowest priority. Execution of this NOP is immediately interrupted by the previously interrupted kernel, its interrupt vector is read, and that value becomes the next address executed.

It was mentioned above that the next address calculated immediately before an interrupt (return address) is *optionally* stored in the interrupt vector for that device. If the option is not invoked (as specified by a bit in the microinstruction word shown in Figure 3), the address is not stored and the return procedure will reinitiate execution at the point previously set in this vector address. This will typically cause re-execution of some instructions. It allows, however, general purpose arithmetic unit registers to be used without fear of their being altered between execution of instructions within a kernel since those instructions will be re-executed if an interrupt does occur. This represents another significant characteristic of the architecture as it reduces both microprogram size and save-restore time. Conventional register-save schemes are inappropriate due to the relatively small register complement in microprocessors, the high interrupt rate (virtually all kernels are executed in response to device requests), and the relatively small size of most kernels.

An interrupt disable bit is included in the microinstruction word for time-critical sequences of instructions. In addition, the priority scheme protects higher speed devices from being delayed by slower ones. Generally, however, it is expected that hard-wired logic will be used to implement time-critical functions utilizing the ease in switching between logic and microinstruction kernels inherent in this architecture.

SPECIFIC IMPLEMENTATION CHARACTERISTICS

Although specific details of the implementation are not critical to the microprocessor architecture presented herein, they are discussed briefly as one example of its utilization. Intel 3000 series components were used to build a prototype system to serve as a display controller for the Tektronix 4014 graphics terminal. Devices connected to the processor include an interface to the 4014, a floppy disk controller, a serial interface to a host computer, and a multiply-divide unit. The system is capable of receiving segmented images over a serial communication line from a host computer, massaging this display data into a compact form, storing it on the disk unit, and displaying the images with three-dimensional translation, rotation, and scaling in both the store and refresh modes of the 4014. Functions performed by the microinstruction kernels ranged from single instruction read character and store in memory to using the multiply-divide unit to multiply four by four matrices for coordinate transformation.

Idiosyncracies of the Intel 3000 components permitted a bit in the microinstruction word, shown in Figure 3, which disables the arithmetic units so that non-destructive tests may be performed. In addition, deficiencies in the conditional branch characteristics required bits specifying branch on carry-in and carry-out.

ADVANTAGES OF THIS ARCHITECTURE

The computer architecture described in the previous sections is intended to provide an approach to simplifying digital logic unit design. Its major feature is the incorporation of a high speed microprocessor to replace portions of the digital logic chain with sequences of programmable microinstructions. When properly applied, this will reduce the package count, wiring complexity, power consumption, and therefore overall system cost.

In applications where timing characteristics permit, multiple devices can draw on a single microprocessor for logic replacement. In addition to the obvious cost and complexity reduction benefits, this significantly simplifies the intra-device communication problems. Scratchpad memory can serve as a buffer to mask the effects of differing device speeds. Internal processor registers can be used to maintain a single copy of interlock and device communication controls. Perhaps most significant, the serial nature of the microprocessor can reduce device race-condition conflicts and serve to isolate the devices for debugging.

Naturally, the usefulness of including a microprocessor in a digital design will depend on the extent to which it can perform logic functions required by the design. Arithmetic and shift operations, common within these processor elements, easily replace hard-wired logic causing significant reduction of space consuming data path wiring. Temporary storage of data and control information, simplified by the processor's internal registers and scratchpad memory, is another candidate for logic replacement. In addition, the processor's ability to control the device sequencing using device control bits and updating the interrupt vector to point to different kernels (states) can significantly increase design flexibility and reduce re-wiring during the debug process.

A final advantage involves simplification of the hard-wired logic debugging. Since, within this architecture, the processor kernels are interacting with the device at various states in the logic chain, test programs can be written to repetitively active isolated portions of the logic. This allows modular debugging and provides repetitive signals necessary for good oscilloscope traces.

A NOTE ON SPEED

In the preceding sections, specific timing characteristics have been avoided as they do not directly affect the architecture presented. The microprocessor system implemented as an example has a basic instruction time of 300-nanoseconds with an additional 200-nanoseconds required for memory reference instructions. That speed limitation is largely due to characteristics of the Intel 3000 elements and implies a minimum interrupt service time of one microsecond. (This includes one memory reference instruction to read the vector address, one non-memory reference instruction branch to the kernel, and execution of the first instruction in the kernel.) This example could therefore sustain megacycle request bursts and approximately 250,000 processor requests per second assuming ten to twenty instruction kernels. This seems adequate for most logic systems, and has proven so for the graphics display processor application.

The real timing issue, however, is not absolute speed but rather the relative speed of the processor compared to available hard-wired logic. Assuming equivalent technologies, the speed difference will depend on the overhead in gate levels necessary to provide multiple functions within the microprocessors. Evidence suggests[1] that this speed reduction (or processor complexity) factor is fifteen to twenty. The microprocessor architecture described herein is oriented toward interleaving kernel execution and hard-wired logic. The speed factor implies that the processor will be usable for those functions in the logic chain where the design timing requirement is at least twenty times slower than the basic logic time necessary to perform the function.

SUMMARY

A microprocessor-based system architecture has been presented for the design of digital devices. It is centered around the interconnection of the microprocessor and digital devices in such a way that various portions of the digital logic chain can be replaced with sequences of microinstructions. Where multiple devices are augmented with a single processor, the architecture provides a very convenient interface between them. A prototype graphics display controller was built using Intel 3000 series microprocessor elements which has demonstrated the viability of the architecture for realistic digital design problems.

REFERENCES

1. Rattner, J., J. C. Cornet and M. E. Holt, Jr., "Bipolar LSI Computing Elements Usher in New Era of Digital Design," *Electronics*, September 1974, pp. 89–96.
2. Bailey, S. J., "Microprocessor: Candidate for Distributed Computing Control," *Control Engineering*, Vol. 21, No. 3, March 1974, pp. 40–44.
3. Hoff, M. E., Jr., "New LSI Components," *6th IEEE Computer Society International Conference Digest*, December 1972, pp. 141–143.
4. Weissberger, A. J., "Distributed Function Microprocessor Architecture," *Computer Design*, November 1974.