Development of a microprocessor support facility for large organizations

by BRUCE E. STOCK
Boeing Aerospace Company
Seattle, Washington

and

MIGUEL A. ULLOA
Tektronix, Incorporated
Beaverton, Oregon

INTRODUCTION

Microprocessor technology has grown in a decade from simple 4-bit controllers to complex 32-bit architectures which rival the performance of mainframe computers. Applications of microprocessors today range from military and aerospace programs to consumer products and toys. The rapid growth of this field has resulted in many different types of microprocessors, based on several different technologies, being offered by a number of different manufacturers. Each of these manufacturers, and several independent companies as well, also provide support tools to aid in the development and checkout of microprocessor-based products. Evaluation kits, development systems, analyzers, high level languages, and software simulators are but a few of the available aids permitting more rapid design, integration, and debugging of systems incorporating microprocessors.

The growth in the number of support tools along with the differences in cost and capability of these tools have created a new set of problems for both management and engineering. It has become difficult to select a set of tools which will provide a level of support appropriate to the complexity of the work being done, while minimizing the cost of these resources. Additionally, the training of personnel, service and maintenance of the tools, and the eventual obsolescence of the resources provided are important issues. The problem is especially acute in large organizations where hundreds of engineers may be involved in the design of a wide variety of microprocessor-based products.

In this paper we will briefly examine the types of support tools available and the suitability of these tools to certain types of development, and finally will focus on a centralized concept suited to large scale development.

MICROPROCESSOR DEVELOPMENT SUPPORT TOOLS

There is a great variety of hardware and software tools to support the development of microprocessor-based products. These tools range widely in their cost, complexity, and capabilities. Low-cost evaluation and learning aids are available to familiarize designers or inexperienced users with particular microprocessors. Development systems ranging from a few thousand to over forty thousand dollars come in different forms and with many options to support various development phases. Development systems from individual chip manufacturers provide support for one or more of that vendor’s family of chips. On the other hand, “universal” systems can support several microprocessors from different manufacturers. To support software development, operating systems, editors, assemblers, compilers, linkers, and debugging tools are provided. For hardware and software integration, support for in-circuit emulation, memory mapping, and real-time analysis is available. To evaluate microprocessors and do software development on mini or large computers, cross-software and simulators are provided by software houses and timesharing networks. These and other tools are outlined in Figure 1.

Despite the variety of support tools, in general only the most popular microprocessors (or those out in the market for some time) are supported by several of these tools. Other chips, including newly introduced ones, are supported only in a limited form. For example, some microprocessors supported by cross-software or low-cost design aids are not supported by development systems or in-circuit emulation. In cases like this, in-house design of special test tools must be considered or other microprocessor choices evaluated.

Another aspect of present support tools to be considered is their suitability and effectiveness in different types of microprocessor development. As a general rule, users of microprocessors are involved in either one-time development, periodic new design, or continuous heavy development. Simple in-house-designed test tools, low cost development systems, and cross-software are often suitable for one-time developments because of their low capital investment. When periodic designs with few engineers are involved, then a good-quality development system capable of supporting several microprocessors can be quite cost-effec-
Evaluation/selection tools
- Evaluation kits, design aids, teaching tools
- Microprocessor development systems
- Cross-software, simulators

Software development Tools
- Microprocessor development systems
- Operating systems, file managers
- Editors, assemblers, compilers, interpreters
- Linkers, library generators
- Simulators, debuggers, cross-software

Hardware development tools
- Test and measurement tools (e.g., oscilloscopes, data generators, pulse generators)
- Logic analyzers, microprocessor analyzers

Software/hardware integration tools
- In-circuit emulators
- I/O simulators
- Memory mapping; real-time analyzers
- PROM programmers

Production support tools
- Diagnostic software
- Microprocessor development systems
- Special-purpose automated test equipment

Figure 1-Microprocessor support tools.

LARGE SCALE MICROPROCESSOR DEVELOPMENT

Large organizations doing extensive microprocessor-based product development have a number of possible alternatives to provide development support. Five basic approaches to be discussed below are stand-alone development systems, stand-alone systems with timesharing, timesharing with remote emulation, multi-user stand-alone development systems, and centralized development facilities. Which approach, or combination of these approaches, will best suit a particular organization, will depend on the organization's present facilities and on its structure and monetary constraints, among other things.

Stand-alone development systems

In this approach, each project or group uses as many individual development systems as is required by the complexity of the task. Each project thus has immediate access to the resources it needs, which include the development system plus the necessary peripheral equipment and test and measurement tools. There are disadvantages with this approach. First, the number of development systems and the total cost increases rapidly as the number of projects increases. Second, resources such as line printers and storage units are duplicated many times, usually without full utilization. And third, knowledge, experience, and software are not easily shared between groups or projects.

Stand-alone systems with timesharing

Here the stand-alone alternative is augmented by using either an in-house timesharing system or one of the national networks to provide additional software development and checkout capability. This approach is suitable for improving software development support without duplication of resources and for including microprocessors not supported by development systems. This is a viable alternative for organizations already committed to a large number of stand-alone systems; for other organizations, the next two alternatives can be more cost-effective. A disadvantage of timesharing is of course the decline in user response time as the system becomes heavily loaded.

Timesharing with remote emulation

With the availability of in-circuit emulation as a peripheral device, it has become possible to avoid the stand-alone development system completely. In this approach, a multi-user computer is connected with emulator systems such as Tektronix 8001 Microprocessor Labs

Multi-user stand-alone systems

Within the last year, stand-alone development systems which support as many as 6 or 8 simultaneous users have become available. Such systems provide the advantages of multiple stand-alone systems while minimizing the duplication of resources such as mass storage and line printers.
These advantages may be offset, however, by problems such as limited line printer capacity, lack of spooling capability, limited storage space, and degradation of user response time.

Centralized development facilities

All the tools necessary for development of microprocessor-based products can be integrated into a physically centralized facility and made available to a large user community as a shared resource, so that microprocessor support is shared and controlled in much the same way as data processing facilities are shared and controlled through computer centers.

The central facility will normally consist of a medium-size or large multi-user computer, mass-storage devices, peripheral equipment, one-line terminals, and hardware/software integration systems. Communication lines for remote access can also be provided. The mainframe computer should include a powerful operating system and standard software utilities plus cross-assemblers, compilers, and simulators to support a variety of microprocessors from different manufacturers. For hardware development and hardware/software integration, the center should have general purpose test equipment, in-circuit emulators, I/O simulators, and PROM programming tools.

In addition to support tools, the center should have an operating staff to provide the following support services: user access control and accountability; user assistance and problem resolution; configuration management and control; system software revision and enhancement; maintenance coordination and consumable resource stocking; coordination of new user training courses and information; and integration of new software and hardware tools to support the changing technology.

The centralized approach provides all the benefits of using timesharing with remote emulation plus additional benefits not realized with any of the other alternatives. In addition to minimizing resource duplication and providing commonality of tools and methods, the center offers other advantages: knowledge, information, and software can be easily shared among all groups and projects; utilization of hardware and software tools is kept at a high level; maintenance and service of tools is the concern of the support staff alone rather than of separate departments; and training for new users is readily available.

The major disadvantage of a centralized facility is its high initial cost but this is offset by maximizing the utilization and productivity of the capital investment in resources. Another disadvantage is the possible decline in system response time when high user demand occurs.

Of the alternatives presented above, using stand-alone development systems is currently one of the most, if not the most, popular approaches followed by both small and large organizations. However, this is the alternative that can least realize all the benefits of centralization, while at the same time it requires the highest overall capital investment. Therefore, it is imperative for large organizations to weigh these alternatives carefully. Boeing Aerospace Company has done just this, and the result of its analysis has been the implementation of the Microprocessor Development Support Center.

MICROPROCESSOR DESIGN SUPPORT CENTER IMPLEMENTATION

System overview

In early 1978 work was begun to design and implement a development facility at the Boeing plant at Kent, Washington, near Seattle. Called the Microprocessor Design Support Center (MDSC), the facility was to provide support to any company project doing microprocessor-based design. The goal of the center's designers was to provide a facility which would: accommodate a large number of users; support a wide spectrum of microprocessors; maximize the usage of resources provided; minimize training and retraining time; use off-the-shelf hardware and software to the maximum extent; and adapt to newer and more complex microprocessors with a minimum of modification.

A centralized design based on a larger minicomputer was selected as the best approach to meeting these goals. This approach was made feasible by the availability of cross-software (cross-assemblers, linking loaders, simulators, etc.) and in-circuit emulation as a peripheral device.

The center is divided into three adjacent areas: a machine room housing the central computer and its related peripherals; a software development room containing CRT terminals in a quiet environment; and an integration room containing the in-circuit emulators, I/O Simulator, and PROM Programmer. Functionally, the computer is the hub of the system, with all other devices acting as peripherals to it.

The MDSC currently provides support for the following microprocessors: 8080, 8085, 8048, 8086, 6800, 6802, 650X, F-8, 3870, 1802, Z-80, Z-800, Z8000, and 68000. Complete hardware and software support is not available for each, but some support, either software or hardware or both, is available for all of the chips named. An in-house development is under way to provide a compiler targetable to several microprocessors. The center also provides software support for bit-slice microprocessor development via a meta assembler on the central computer. A Control Store Simulator is being acquired which will permit download of assembled microcode to the users' bit-slice prototype as well as debug and trace capabilities.

Computer

The primary requirement for the central computer and its immediate peripherals is to support a timesharing environment with a large number of simultaneous users while providing a reasonable response time to each user. To satisfy this requirement, a PDP 11/70 has been selected and equipped with a megabyte of local storage, 356 megabytes of rotating storage, 48 RS-232 ports, and a DMA channel. The installation also contains the usual magnetic tape, line
printer and card reader peripherals. A rack mounted modem set with 16 channels (12 currently active) is mounted within the main computer cabinetry to give dial-in users access to the system.

Software

The top level of software on the facility is the operating system. This package permits the computer operator to configure the system and allocate resources and priorities to tasks. At the user level, it provides access to all of the installed utilities of the system.

The MDSC operating system is the Interactive Application System (IAS) provided by the computer’s manufacturer. This system is well suited to the MDSC environment in that it provides concurrent running of realtime, timesharing, and batch jobs. This capability permits a user at a terminal to submit a long assembly or other job to the batch stream and then continue to edit or debug other modules while it is being processed. Concurrently, another user may be exercising a real-time I/O simulation task. The operating system provides access to an editor, file management utilities, word processing software, compilers, cross software, and a library of user-generated code modules. In addition there are three major in-house designed programs on the system: an Emulator driver, an I/O simulator driver, and a PROM programmer driver. The Emulator driver permits interactive communications with the in-circuit emulators, transfer of developed code to and from the prototypes and several enhancements to the basic command structure of the emulator. The I/O simulation driver provides an interactive control package which permits a user to generate data bases or data streams, to output them with selectable rates and protocols to his prototype, and to collect and display the response to them by his prototype. The structure of the program also permits closing the loop to provide a limited environmental simulation; that is, the responses from the prototype may be used to modify or generate the data stream going to the prototype. The PROM programmer driver permits interactive control of the entire PROM programming and verification process, thereby minimizing the opportunity for human error. In addition, utilities are provided to partition object code modules of arbitrary word width and length into ROM-sized modules, and to provide object output media in formats compatible with production-line programming equipment.

Emulation

Perhaps the greatest single aid to microprocessor integration and checkout is the technique of in-circuit emulation. An in-circuit emulator provides for control of and visibility into the execution of the user’s software on the actual prototype hardware. By plugging into the CPU socket, the emulator affords these advantages without any special interface wiring or hardware required in the prototype.

Ideally, an in-circuit emulator would provide a perfect replica of the operation of the emulated chip. In practice, however, in-circuit emulators tend to suffer from one or more faults, such as:

Extra delay

The use of buffers on most, if not all, of the probe pins and the inevitable delays due to extra cable length between the emulator and the prototype causes some timing differences. These are usually quite minor and only become significant for the fastest chips when employed in designs with tight timing.

Non-realtime operation

Some timing and interaction problems only manifest themselves when the prototype is running at full design speed. Other designs may in fact not run at all in less than realtime because certain critical tasks cannot be serviced quickly enough. Some emulators have internal delays which require wait states to be added to the microprocessor’s cycles. These wait states perturb critical timing tasks and mask marginal access time problems. Other emulators may permit the prototype to run at full speed, but are then unable to provide any insight into what is happening in the prototype (i.e., no trace information).

Designed-in differences

Since the emulator must perform additional tasks in the course of its operation (such as pausing and dumping register contents to a CRT) which the actual microprocessor will not have to cope with, the emulator designer must make some decisions as to what will be presented to the prototype CPU interface during these operations. Depending upon how these questions are answered, the prototype may experience a loss of refresh signals, unexpected pulses on control lines, unexpected tri-stating of lines, or other unwanted anomalies.

Expropriations

The design of the in-circuit emulator can result in its requiring part of the prototype memory or I/O address space. In other cases, a major interrupt line (such as the non-maskable interrupt) may be used by the emulator, requiring the user to work around the difficulty in his prototype.

Different fan-in/fan-out

The use of TTL buffers at the probe causes different fan-in/fan-out characteristics from the actual MOS, CMOS or bipolar processor. This can result in learning late in the integration cycle that the prototype functions perfectly with the probe in, but not with the actual microprocessor.

The MDSC has selected the Tektronix 8001 Microprocessor Development Aid for use as its standard in-circuit em-
ulator. The 8001 is designed specifically to be used in conjunction with a host system which provides software generation capability. It is possible to configure the 8001 to emulate any of several different microprocessors by inserting appropriate emulator-board/probe units. By using a common emulator mainframe such as the 8001, it is possible for a user who has been working with an 8085 based design, for example, to move on to a Z80 project with his entire emulator command repertoire and debug strategy intact, representing a significant reduction in retraining. It is also possible to move optional equipment resources (extra memory, Real Time Trace Modules, etc.) from one user who does not need them to another who does.

With this configuration, the 8001 is integrated into a "super development system." A user at a terminal on this expanded system can model system performance using a High Level Language, perform trade studies and run benchmarks on several different microprocessors using the simulators, and prepare trade study and program documentation. He can then generate software for any of several different microprocessors, down load it to an 8001, and exercise his prototype with that software. At any time he can return to the Edit/Assemble mode to correct errors, and continue with the modified code. This process is repeated until the prototype is functioning correctly. The software is then committed to ROM, the microprocessor chip is replaced in its socket, and the prototype is released for validation/verification.

**PROM programming**

A centralized developmental laboratory such as the MDSC does a considerable amount of PROM programming in support of the integration effort. To accommodate the wide variety of devices used, a System 19 Programmer made by the DATA I/O Corporation of Issaquah, Washington, was selected. This unit accommodates a large number of programmable devices via plug-in personality modules and has a well developed RS-232 interface to a host computer. The remote control capability via this interface permits the operation of the programmer to be controlled by an interactive software package on the host computer.

The wide variety of devices which can be programmed by the System 19 exceeds the capability of the facilities provided on stand-alone microprocessor development systems. Further, a large number of down-load data formats can be accommodated, such as Binary, Intel, Tektronix-hexadecimal, ASCII, and others.

**I/O simulation**

The MDSC provides an I/O simulation device called the Adaptive Interface Unit (AIU) which is installed at a specific integration station. The purpose of the AIU is to provide the user with a generalized hardware interface to his prototype over which user-defined stimuli may be provided and the prototype's responses can be obtained and evaluated. Used in conjunction with in-circuit emulation, I/O simulation provides an effective means of verifying correct prototype operation prior to leaving the lab. The characteristics of the AIU/user interface are as follows:

**High speed parallel I/O**

A 32-bit-wide input channel and a 32-bit-wide output channel are provided with local buffering of 4K words on each channel. Transfer rates to 1 megaword per second can be accommodated with user-clocked, AIU-clocked or handshake data transfer protocols. Differential TTL drivers and receivers are provided, although special drivers or receivers can be provided on a standard plugin card.

**Low speed parallel I/O**

A 16-bit-wide input channel and a 16-bit-wide output channel are provided with rates to 1000 words per second. In addition to strobed and handshake protocols, there are modes which provide for output of a predefined data stream at predefined intervals and input of data with associated time tags. Differential TTL drivers and receivers are provided.

**Serial I/O**

Serial data at rates up to 10 megabits per second can be accommodated with or without embedded sync patterns. User clocking or internal clocking of data can be accommodated.

Obviously the AIU cannot be directly connected to all the different prototypes that come into the MDSC. However, a level of generality has been designed into the AIU/user interface which permits interfacing most equipment with no additional hardware.

**Special test equipment**

The MDSC maintains a set of special test units which are used to verify the operation of the emulator probes. They are used whenever a malfunction cannot be immediately isolated to either the prototype or the emulator hardware. Each test unit is built around a single board microcomputer to which is added a power supply, LED displays and a small amount of additional discrete logic. The function of the box is to verify correct operation of the emulator probe to a certain level of confidence. This is done by executing a ROM resident program which requires correct operation of the emulator address, data, and control lines in order to run successfully. The test units provide a necessary standard of operational reference and act as a test device to verify emulator performance before initial use or at any time thereafter, if a malfunction of the emulator is suspected.
Costs

The total cost of procured hardware and software for the basic MDSC configuration was $411,736. This figure includes the cost of the computer and its peripherals emulation hardware, cross software, modems, terminals and PROM programming equipment. It does not include the cost of the AIU, in-house-designed software, and general purpose test equipment. There are now commercially available versions of some of the software that was designed in-house (The Emulator driver and the PROM programmer driver). Purchase of this software would have brought the overall cost up to approximately $417,000. In its current form the MDSC is supporting over 130 active user accounts, representing 25 different project groups.

SUMMARY

Given the ever increasing use of microprocessors and the dynamic nature of the state of the art, there is a need to optimize developmental and support strategies so that capital investment, retraining of personnel, and obsolescence of equipment are minimized. Computer facilities have been centralized to advantage in many organizations to provide a range of resources to a large user community. Similarly, development tools for microprocessor support can be centralized to advantage.

Boeing’s Microprocessor Design Support Center has proven to be a cost-effective way to provide integrated microprocessor development resources in a large-scale applications environment. The centralized facility resulted in significant savings when compared to the estimated $649,000 cost of acquiring decentralized stand-alone systems. In addition, there are substantial benefits in efficiency, adaptability to the changing technology, easier and less costly training and, in general, more time available for direct product development.

REFERENCES