:MMM. 0[illegible]/16/69 0952:25 JFR ; .PLO=1; .RTJ=0; .PGN=0; .ROM=1; .DSN=1;
.LSP=0;.HED="MULTI-MINI-MACHINE"; .DPR=0;

ABSTRACT

This working document describes a method of extending the AHI-NLS facility on the SDS 940 from 12 to 32 consoles.

FOREWORD

Purpose

This memo discusses the usefulness of small computers as special-purpose hardware device controllers and as complex, programmed feedback mechanisms.

Timings and simulations indicate that the current level of SDS 940 response and display-oriented feedback can be maintained, and maybe improved, while the number of consoles is tripled.

The centralization of the display buffers and the use of small computers as display controllers allows experimentation with ring-structuring techniques for inter-console collaboration.

Similarly, the use of small computers as input device controllers contributes to collaboration by providing an elegant means for linking an arbitrary collection of input devices to any console.

Finally, the small machines can quickly react to most of the single-character NLS interactions, reducing the frequency of 940 computation per user and thus reducing swapping.

History

This memo is the result of discussions held in late December 1968 and early January 1969.

The main contributors have been Roger Bates in hardware and Don Andrews, Bill Paxton, and Jeff Rulifson in software.

GOALS

The initial study of 940-NLS operating characteristics centered around the goal of operating 12 work stations with response equivalent to the 6 currently operating.

Bottlenecks in the 940 system are connected with core space and swapping.

Currently, displays are stored in 940 core. Each user has his own buffers, and a new 940 core page is frozen as each user enters the system. For 5 or 6 users this is not too bad, but for more than 7 or 8 it is totally unreasonable. see(LOADS,Conclusions)

The immediate feedback that NLS gives to each user is significantly more complicated than that which can be offered through conventional echo tables.

Consequently, each NLS user must be immediately activated for each character or button push.

Statistics on NLS usage indicate that if the display buffers are removed from 940 memory, approximately 10 to 12 work stations can be serviced before the swapping reaches a critical point, and response significantly declines. see(LOADS,Conclusions)

The most direct solution for the 12-station system is simply a bulk core for display buffers. This approach has already been discussed in a current proposal to RADC for system expansion.

Our statistical observations indicate that the 940 could serve approximately 32 NLS users, if users interacted with the 940 on a work-request basis instead of a single-character basis. see (LOADS,Usage)

When more than 12 work stations are considered, the single-character interaction with the time-sharing system and user programs completely overloads the system.

The first criterion for any expansion is that it be extendable to the limits of the 940.

The second is that the system closely resemble one that may be expandable to even larger service, say 100 consoles.

If such expansion is flexible and modular in its design and hardware coupling, then alternative programming, scheduling, and queueing techniques may be tried. The problems of much larger systems may be incisively studied and the solutions tested through implementation.

MULTI-MINI-MACHINE

It has become apparent that collaboration, in its many forms, must be implemented in NLS.

Three possible modes are:

Display-frame sharing, where different logical parts of a user's current display picture may be viewed by a set of users

Input-device flexibility, where many people may sit around a table, each with his own handset and mouse; or users remote from each other can drive one another's displays

Audio communication, which takes two forms:

Complete voice switching, so that any number of "conference calls" could be established between the work-station users

Computer-generated sound, used as feedback to a single user or a group of users in collaboration.

Many of these features can eventually be implemented totally entirely within the framework of the 940. However, the current hardware/software approach imposes severe limitations in two ways.

The MONITOR in the 940 time-sharing system is cramped for space. This means that the extensive tables necessary for input-device interchangeability will not fit, and limited, costly schemes would have to be implemented.

The addition of hardware devices is governed by the availability of special-purpose controller and polling devices.

Each of these operates a fixed number of each kind of device.

We would prefer a scheme that allows new devices to be added and interchanged freely and in small increments.

Other Devices

What about the printer?

We might want to move our printer to a small machine.

What about micro-film output?

We might want to drive a microfilm machine from a small machine.

What about Model 37 TTYs?

What about the NET?

The idea is that the big core might serve as a buffer for messages from the IMP.

However, we will wait until more is known about the IMP before we do any planning.

What about the NIC?

The idea here is that small machine could handle many NIC users if they ran a dedicated information-retrieval/editing system. The small machines could be a single user on the 940, again reducing the swapping effort while maintaining high-speed response.

What about storage tubes?

What about an on-line Dura?

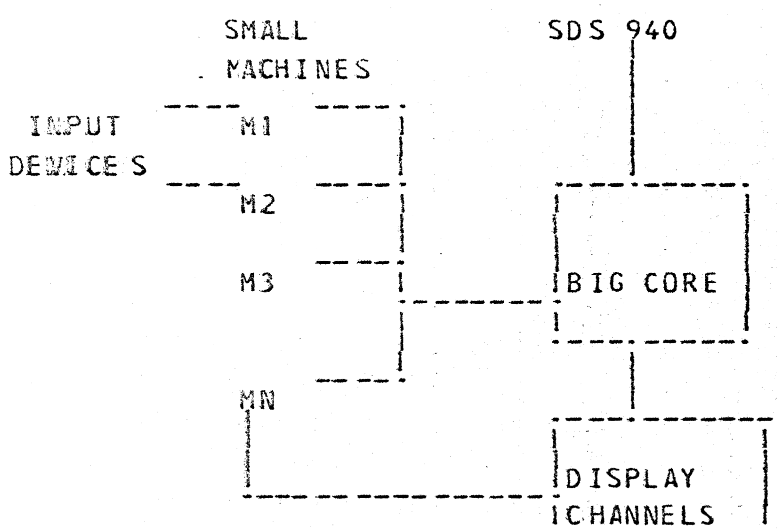
THE MULTI-MINI-MACHINE APPROACH

General Idea

There is a big core of 16-bit words. Most of it is used to store the ring of all the display buffers. Some is used to store queues of tasks, tables for associating users and input devices, and reentrant interpretive code.

Connected to the core is a bank of small machines (4K 16-bit words, 4 usec instruction times). They are used as input-device controllers, Interpretive code executers for complex feedback, and channel drivers for the display channels.

Gross View



The small computers more than meet the minimal time requirements.

Each small machine has its own small core memory. If it is operating on the ring structure and determining feedback actions, it contains an interpreter for reentrant code which resides in the big core.

This code is close to the code which is compiled from the Special-Purpose Languages (SPLs) for the current NLS.

The code is not algorithmically complicated. Each input character involves only a few logical decision and table changes.

The decisons, however, may involve reference to the user's ring of display buffers, and the few hundred words necessary to define his current status.

It is this large amount of information that causes the swapping overload in the 940.

By having all of this information immediately accessible (in core rather than on drum), a small slow machine is more than adequate.

Many small machines were chosen over fewer faster ones because they could be manipulated and added in small increments.

Input devices, along with the controlling processors, can be added a few at a time as different experiments arise.

Since the logic for the feedback is in the big core, it may be expanded to two or even four time the orginal estimate. Thus the small machines will not be cramped for space.

The structure of the display rings can be modified as information is gained on the nature of display collaboration. This would never be possible if the display controller were built into fixed hardware.

If many machines are prepared to perform the same task, some simple queuing and access methods result in better machine utilization and small changes in system performance as users are added.

The queues are all in the big core. There are two basic kinds:

As characters come in to the dedicated input-controller machines, they are associated with logical users and deposited on input queues.

When they are free, the feedback machines poll these queues.

When characters are found, they are processed.

Sometimes the 940 must be requested to intercede and complete a task. When this is so, the logical user is put in a state of limbo as far as the feedback processors are concerneds and the task is queued for the 940.

A simple implementation of the "Dijkstra Flag" keeps two processors from simultaneously modifying a data structure.

All the small machines share a single access line to the big core. Thus only one can be doing a read or write on a single cycle.

There are, however, three modes of memory access: read, write, and Dijkstra read. In the latter a word is read from memory and sent to the small machine. During the write-back in the big core, however, the top bit is cleared.

Suppose that a word in the big core is a pointer to a queue and that if the top bit is on, the queue is free; otherwise a processor is working on the ququе.

If a processor Dijkstra-reads the pointer and the queue is free, it then has control of the queue as well as the pointer to it.

If the queue is in use, however, the processor is informed of this.

The 940 must have two kinds of control over the small machines.

There is a two-way interrupt line.

This permits the small machine to notify the 940 that there is something on a queue.

It also permits the 940 to interrupt the small machine and request, from a small resident program, sufficient information to run a real-time debugging package.

It is also necessary for the 940 to exercise complete control at certain times (e.g. system startup or a looping small machine). To do this the 940 must be able to operate the console switches, and read most of the console lignts.

There are two alternative methods of driving the displays on the system.

A small machine could drive the display directly. This would require the machine to output a word to the display every 15 usec. Only the fastest (and most expensive) of the small computers are capable of doing this. Even then they barely make it, and cutting it this close seems inadvisable.

The other approach is to have a data channel which can be loaded and activated by the small machine. In this way the small machine still interprets the ring structure and thus maintains the flexibility there. Using channels, the small machine can be relatively slow and still drive the displays at full capacity.

Two auxilary ideas are necessary to make this clear.

A channel normally contains an address register and a word count. These channels also contain an address-register buffer and a word-count buffer.

As the channel is driving the display from the working registers, the small machine can be loading the buffer registers.

When the channel is done, it issues an interrupt to the small machine. As soon as the machine has figured out where the interrupt came from, it can pulse the channel. The channel can then transfer the buffer registers to the working register and begin driving the displays again.

This cuts the time between buffers from 60 usec to under 20 usec.

The small machine can still have problems keeping up with the display channel if the buffers are displayed faster than the ring can be searched in the big core.

To overcome this, the small machine only searches a ring when it is changed. As it searches a simple list of necessary buffers in made it the small machine local memory. Thus the next channel address and word count are readily available within the immediate addressing structure of the small machine.

These machines can keep close watch over their time allocation. Single overloading of a display causes flicker only on that console; an extensive ring structure affects only the consoles displaying from it; and in general, bad side effects which now propagate from console to console are confined.

Input Devices

Special Input Devices

Each device will be read as a source of 16-bit code. However, the full 16 bits will not be used. Each device will look the same to the input machine; its existence and type will be defined through tables which can be set up and modified by the 940.

The devices planned for are:

A keyboard which produces an 8-bit code

Up to 16 pushbuttons, although we will start with only 8

The mouse, which requires two 10-bit numbers as coordinates, and will thus take two words (one coordinate in each).

Each group of 24 input devices will have a special polling mechanism which works in the following way.

The polling may be started and stopped by the small machine.

When the polling is in operation, it operates continuously at its own rate. This will probably be set so that each device is sampled at least once every 5 to 15 ms.

As each device is sampled, its contents are read into a core location in the small machine.

If the device is a keyboard, however, the core location is changed only if the strobe is on.

The input machine may then inspect the list in core at a slower rate, say every 30 ms, and notice any changes.

The following are the maximum input rates for each device. While it is unlikely that all devices will operate at maximum rate for an extended period of time (e.g. one second), we must plan for at least half-second periods of maximum transfer rates.

When a mouse is in motion, it is usually sufficient to update the tracking display buffer every 30 ms. The coordinates must be observed at this rate, but they can be compared in the input machine, and if the mouse has not changed position the coordinates are not associated with a logical user nor are the display buffers changed.

The maximum rate for a keyboard is slightly under 100 ms per character. The keyboards are polled as fast as the mouse. However, the character in the core of the small machine is changed only when the strobe is on. Thus the small machine has at least 50 ms. to notice and process any character. Every character must be associated with a logical user and deposited on the appropriate queue in the big core, along with the time and the mouse position.

The pushbuttons are polled just like the mouse. Every 30 ms, the contents must be observed by the input machine. When any change is noted, extra action is required.

If the buttons are on the mouse, the up or down action must be associated with a user and deposited on a queue in big core along with the time and the mouse position.

If the buttons are on the handset, changes are ORed together until all switches return to the UP status. The chord is then associated with the user and put on the queue along with the time and the mouse position.

Output Devices

There will be only two high-speed output devices.

Each display channel is loaded from a small machine and transfers words from the big core to 6 displays. Each channel has a 20-bit address register, a 16-bit word count, and an interrupt line back to the small machine. The interrupt is raised when the word count goes to zero.

The audio switching system is viewed as a big switching matrix from the small machine.

- The inputs are all the voice lines and all the tone generators. Each tone is fed in as three separate lines of amplitudes 1, 2 and 4.

- The outputs from the matrix go directly to the speakers or earphones.

- The audio system is used at a ms rate when the tones are in a decay state (making them much more pleasant to listen to).

The following are the maximum rates for the two devices. These rates may be common, and plenty of leeway must be allowed.

When the display channels are running they do not effect the small machines. Thus the times which concern the small machines are the length of time it takes to load a channel and the average length of time to process a buffer once the channel has been started.

We expect the displays to write a character in 7 usec, so the channels will transfer a word from big core to the displays every 14 usec.

Buffers can be a minimum of 1 word long, and thus be processed in 14 usec. Normally, however, they are no shorter than 4 words, which takes 56 usec.

To keep the displays running wide open, the small machines should make every effort to minimize the time to load a buffer. Using the scheme discussed earlier, this time can be cut to less than 20 usec.

If the displays are storage tubes, the buffers will be the character strokes, and these will be stored in the big core.

In this case the small machine will have to look at the buffer generated by the 940, take it apart character by character, and drive the display with the individual character buffers.

Since the storage tubes take about 100 usec to write a full character, this will not overload the small machine, although it will occupy almost all of its time.

Voice switching may be a slow process. Tone decay, however, must be rapid.

Extra I/O Devices

Local Dura?

Storage Tubes?

Potter Printer?

MULTI-MINI-MACHINE

Model 37 TTYs?

Microfilm Output??

Steps for Total Input/Feedback Service

The following is an outline of the steps an input machine will go through as it gets characters from the input devices and puts them into the appropriate queues in the big core. This outline was used to estimate the total processing capacity necessary in the input machines.

Check queue for transfers to big core.

Get character from internal queue.

Make correspondence between hardware device and logical user.

Get queue control for logical user.

Put character on logical user in big core.

Make up appropriate, internal audio queue from logical user information in big core.

Give up queue control for logical user.

Get clock interrupt.

Go down input character buffer.

If same as last interrupt do nothing.

Otherwise

If keyboard, note time and mouse position, put on queue to go to big core.

If pushbuttons, decide which:

If mouse, note time and position, put on queue to go to big core.

If handset, then

If all are up, this is a character end so note time and position and despoit on queue for big core.

Otherwise, build up chord by ORing in new down buttons.

Audio Queues

Do appropriate amplitude switching for computer-generated sound decay, remove processed entries from queue.

Date/Time Queues

Some of the small machine will devote almost all of their time to searching the ring structures and driving the display channels.

Get display channel interrupt.

Use local list with update when ring changes, as discussed above.

Have all the entries in local memory, just put it up.

Changes are flagged, and require a block transfer of new pointers.

The rest of the small machines will be devoting all of their time to processing the input queues prepared by the input machines, updating display buffers, and preparing queues for the 940. The following outline of their duties was used to estimate the processing capacity necessary for the job.

Check for work to do.

Locate non-empty logical-user task queue.

Get IFM control for logical user.

Input character for processing.

Get user state.

Begin execution of interpretive code.

Get character from queue.

Get input-queue control for logical user.

Get character off queue.

Give up input-queue control.

Step through main control.

Update appropriate display buffers.

How is this done if we have long buffers? Is there a display Dijkstra flag?

There is also the problem of writing in the mouse coordinates while they are being changed.

Continue until either

Queue is empty:

Put him to sleep.

Or 940 is needed:

Put him on 940 queue.

Set flag for 940 interrupt.

Update date and time

Give up IFM control for logical user

Interrupt the 940 if necessary

PROPOSED HARDWARE

Data Flow

Interrupt Lines

(pgb) Put-Get Box

- (adr) Address register, 20 bits
- (tr) Transfer register, 16 bits
- (fc) Function code
  - Read
  - Write
  - Dijkstra test
  - Automatic increment

(ch) Channel

- (cadrb) Channel address-register buffer, 20 bits
- (cwcb) Channel word-count buffer, 16 bits
- (cadr) Address register, 20 bits
- (cwc) Word count, 16 bits

(cc) Core controller

- Two-way channel

(cs) Console switches using the NOVA as an example

- Two 16 bit NOVA readout registers--data and PC
- Sixteen bits of input switches
- Readout of extra minor-cycle console lights?
- Single carry bit
- Eighteen functions to select

PROGRAMMING TECHNIQUE

The 940 plays an important role in both the programming and the running of the small machines.

From a programming point of view, everything can be done on the 940.

The interpreter for the small machine can be coupled with good DDT to give debugging aids far beyond those that could be provided by the small machine itself.

Not only can checkout begin before the hardware is ready, but programs can be changed and debugged while the hardware is in use.

Moreover, the file system of the 940 is available to the 940 interpreter and DDT, so no new file system has to be created.

The 940 also has the ability to operate the console switches of each small machine. This will be used in at least three ways:

During initial startup the 940 can bootstrap-load the machines.

While the machines are running, they can be debugged from the 940. At any time the small machine can be stopped, examined, and restarted on a usec basis. The 940 can then take the information and, using symbol tables from assemblies and compilations, provide a monitor and debugging service.

Accurate statistics may be gathered about the operation of the machine. With control of the consoles, the 940 could even gather samplings of the instruction-counter contents, something which is normally quite difficult.

All the programming for the small machine will be done in a Machine Oriented Language similiar to the one in use on the 940 (MOL940). The compiler will be written in Tree Meta, and should present no special dificultiies.

The interpreter and DDT for the small machines will be written in MOL940, and operate on the 940. Since the DDT is executing through an interpreter, many fancy features can be added, such as operand fetch breakpoints, memory reference breakpoints, automatic timing statistics, etc. There will be two modes of operation for the DDT.

One is the normal mode of executing the code with the interpreter.

The other is the SYSdebug mode, where a live, running machine is debugged. In this case many of the fancy features may not work (like memory reference breakpoint while in run mode).

OTHER APPROACHES

Only big core

- single character interaction problem

Single large external machine

- sufficiently large core comes only with big cpu's, it fact too big.
- no fail safe features

fewer faster mini-machines

- growth comes in too large increments
- halfing the speed does not seem to double the capacity

All machines using single core

- poor addressing structure of small machines for huge memory
  - considered memory map it tradition of ATLAS--too many core cycles
  - simple relabeling cause trouble in display rings
- requires too many memory ports to keep small machine running efficiently