Interrupt processing with queued content-addressable memories

by JERRY D. ERWIN and E. DOUGLAS JENSEN
Southern Methodist University
Dallas, Texas

INTRODUCTION

One of the most significant problems in designing high performance computing systems is the complexity of the associated supervisory software. This is especially true in multi-user environments: the software overhead involved in user communications and resource allocation normally absorbs a great percentage of the system's computing power.

An often-proposed solution to this problem is to remove some of the time-consuming executive functions from the software and perform them in hardware. Numerous examples of this operation are visible throughout the history of computer development. These attempts have met with varying degrees of success because as the tasks to be transplanted become more and more comprehensive, it becomes less and less obvious exactly what sort of hardware structures are needed for their efficient implementation.

One of the most vital ingredients in an on-line computer is a powerful and flexible priority interrupt system. More than any other single feature, the interrupt structure determines the capability of the machine to respond quickly to both internal and external stimuli. Whether the computer is used for control or data processing, its effectiveness is frequently measured by how rapidly it is able to react to conditions in the user environment.

A fundamental characteristic of an interrupt system is how many interrupting sources it can handle. Few of today’s computers are designed to accommodate interrupts from a large number (say hundreds) of devices, despite the growing requirements for such facilities. It is increasingly common to find cases where a machine’s processing power is sufficient to satisfy a great many concurrent demands, but its interrupt system is incapable of supporting the corresponding volume of service requests. This limitation lies not only in the hardware restrictions on the number of interrupt lines, but also in the software’s inability to effectively deal with the interrupts it does get. It is apparent, then, that in such cases a brute force extension of current interrupt concepts is inadequate. Instead, a new approach needs to be formulated which provides improved performance from both hardware and software.

It is well recognized that on-line computers generally cannot afford the inefficiencies inherent in scanned or multiplexed interrupt structures; a multi-level hierarchical system is preferable for such applications. Frequently the on-line interrupts do not all have distinct priorities, so they can be separated into priority classes. Each class can then be assigned to some priority level which is shared by all interrupts of that class. Unfortunately, most machines do not possess an efficient means of identifying different interrupt sources within a priority level, and so the user must either degrade his system performance or undergo the expense of additional priority level hardware.

The assignment of priority levels to particular functions can be a complex task since the interrupt requests must be ordered on the basis of their interaction, not merely on their relative importance. However, it is difficult to accurately forecast the exact nature of these interactions in advance, especially since the context tends to vary widely during system operation. The resulting assignment compromises can be avoided by supplying the freedom to dynamically reallocate priority levels, a powerful tool which enables the executive software to accurately establish the computer’s response to changes in its environment. This capability is currently approached through some combination of arm/disarm commands, program-generated interrupts, multiple level assignments, and hardware switching matrices.

One of the disabilities of conventional priority
interrupt schemes is that if an interrupt at some level is waiting, or is being serviced, or is suspended due to higher priority requests, subsequent interrupts at that same level will be ignored. It is obviously necessary for a system to be designed so that the time required to respond to an interrupt is less than the average time between occurrences of that interrupt. But it is also highly desirable that the system not become saturated by occasional bursts of interrupts at any level. This tolerance to interrupt asynchronism greatly eases the problem of compatible priority assignments, and lessens the risk of disrupted communications between the machine and its environment.

These considerations have all been successfully addressed in the design of a special purpose Interrupt Processor (IP). The IP functions as an autonomous element of medium to large scale on-line multiprocessing systems, and facilitates the computer's interaction with users, peripherals, and control interfaces. All of the functions associated with detecting, acknowledging, and scheduling interrupts have been incorporated into the IP hardware, providing centralized routing of interrupts to all other processors. The remainder of the task assignment mechanism may be software in the destination processor\(^4\) or a hardware extension of the IP.\(^5\)

The IP monitors a large number of interrupt lines on which signals are automatically identified by source within software-defined priority levels. When interrupts are received they are organized and stored on the basis of priority and order of occurrence. This assures optimum priority response, and reduces supervisory software overhead leaving more processing power for the users.

The IP is organized around a special unit called a queued content-addressable memory, which forms its primary storage and processing facility.

THE QUEUED CONTENT-ADDRESSABLE MEMORY

The concept of a queue is a familiar one to hardware and software designers alike. As modeled in software, queues are usually variable length, each new word being added directly behind the last one. Hardware queues, on the other hand, are fixed length as exemplified by a special shift register in which every new entry is inserted at one end from where it moves up to the forward-most empty position.

Content-addressable memories (CAM's) are also well-known, but not so widely found, due to technological limitations which are now being overcome.

Queues and CAM's are not necessarily disjoint structures, but can be combined into one entity having both kinds of characteristics. As shown in Figure 1, this results in a wordwise 2-dimensional memory consisting of a CAM with a queue behind every word. To enter a new item into the memory, the fronts of the queues (i.e., the CAM words) are associatively searched for a current entry having a key that matches that of the new item. If there is none, the new word is placed directly in a vacant CAM slot. If a word with a matching key is found in the front of some queue, the new item is entered into the rear of that queue and allowed to ripple forward. To read from the memory, the CAM is interrogated for a word with the desired key. If one is present, it appears on the output lines. If an entry is removed, the remaining words in the corresponding queue all move forward one position, filling the vacated CAM position.

INTERRUPT PROCESSOR ORGANIZATION

For purposes of discussion, it is assumed that the IP is part of a 32-bit computer, and that it interfaces with both the Central Processor (CP) and the main memory. Only one CP is mentioned, but the IP is readily adapted to multiple CP's in the fashion described in References 3 and 4, or 5.

The basic IP configuration shown here provides for 64 levels of priority with 16 hardware-identified sources per level, for a total of 1024 interrupts. This may be
expanded in increments of 64 levels to a maximum of 256 levels and 4096 interrupt sources.

A block diagram of the IP is illustrated in Figure 2. The major components are a priority structure, a random access scratchpad, and a queued CAM.

Priority logic

A fully nested priority tree monitors the 64 to 256 interrupt lines. This tree determines the order in which interrupts are accepted and stored for processing, as contrasted with the program-controlled priority in which the stored interrupts are serviced by the CP.

The priority logic incorporates a holding register that frees the device from having to maintain the interrupt signal. Also included is an arm register with a bit for each line which may be set and reset both unitarily and in groups.

The highest priority line which is armed and active is encoded by the tree into an 8-bit binary number. This number is transferred to the Scratchpad Address Register (SAR), and the corresponding 4-bit source identification code is stored in part of the queued CAM input data register. A latch is then set to return an acknowledge signal to the interrupting device. This latch also removes the interrupt line from the priority tree until the interrupt is reset by the device.

Scratchpad

The scratchpad is a high speed random access memory which contains one word for each of the 64 to 256 interrupt lines. The words are 25 bits long and formatted as follows:

```
LEVEL  POINTER  E
0       8        24
```

The 8-bit level field indicates the priority assigned to the associated interrupt line, and can be altered under program control. More than one interrupt line can be placed on the same level if desired, and under certain conditions the waiting requests from a given line may have more than one priority.

The 16-bit pointer is used by the CP to form the address of the appropriate interrupt service routine. This frees the computer from having fixed interrupt entry locations in main memory.

The E bit is both a unitary and group enable bit. Considered as part of a scratchpad word, it can be set or reset unitarily. However, the scratchpad is also sideways addressable in its least significant bit position. Viewed together as a vertical word, the 64 to 256 E bits are divided into 8-bit groups. Each group may be set to all ones or reset to all zeros according to the values of the corresponding bits in a 32-bit word supplied by a CP instruction. Interrupt requests received on a line which is armed but disabled will be accepted and queued up, but will not be processed until the line is enabled.

Scratchpad read requests come from either the priority logic or the queued CAM, with preference given to the latter in the event of conflict. In all cases, the scratchpad is addressed by line number.

There are two means of altering the scratchpad's contents (in addition to the sideways addressability of all the E bits). A single word may be replaced, or any sequence of contiguous entries may be reloaded from an image in main memory. In either case, only one CP instruction execution is required. During block updates, the boundaries of the affected area are defined by a pair of 8-bit upper (ULR) and lower (LLR) limit registers. Once a transfer commences, the upper address remains fixed while the lower address is incremented as each new word is written. A comparator not only detects the end of the operation but also monitors the scratchpad address register. All attempts by the priority logic and queued CAM to reference within the boundaries of the limit registers are subject to restrictions specified in the CP instruction which initiated the scratchpad modification. The scratchpad may be copied into main memory without imposing access restrictions on the priority logic and queued CAM.

Queued CAM

The heart of the IP is a queued CAM, as illustrated in Figure 1. It contains at least eight words of content-
addressable memory, expandable in increments of four words to a maximum of 64. Every CAM word is backed up by an 8-word queue, each of which is individually expandable to 32 words in groups of eight.

Each CAM word and its affiliated queue is dedicated to a particular interrupt level as long as that level is armed and has an active or waiting interrupt request. Multiple interrupts on a single level, whether from the same or different sources, are lined up in the queue for that level in order of occurrence. Thus, the CAM size represents the maximum possible number of simultaneously active levels, and the length of a given queue represents the maximum possible number of simultaneously active sources at that level. The relationships between these figures and the number of implemented interrupt levels are parameters dependent on the specific system application and performance requirements.

Each queue entry is 21 bits wide with the following format:

<table>
<thead>
<tr>
<th>LEVEL</th>
<th>LINE</th>
<th>ID</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td>16</td>
<td>20</td>
</tr>
</tbody>
</table>

The priority level is obtained from the scratchpad; the line number and source identification (ID) are received from the priority logic. The F (Filled) bit is set if the word contains meaningful data, and is the mechanism by which entries are automatically shifted forward in the queue. When a word is accessed in the CAM, it may remain there or it may be deleted by resetting its F bit.

The CAM entries are 23 bits wide, as follows:

<table>
<thead>
<tr>
<th>RE</th>
<th>LEVEL</th>
<th>LINE</th>
<th>ID</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>10</td>
<td>18</td>
<td>22</td>
</tr>
</tbody>
</table>

The R (Running) bit being set signifies that the service routine for this level has been activated in the CP. The E (Enable) bit in the CAM is updated as each new request enters the CAM and whenever its counterpart changes in the scratchpad. The other fields are the same as in the queue.

Almost all searches in the CAM are conducted on the level field (or the combined E and level fields), which eliminates the possibility of multiple matches since CAM entries are uniquely allotted on the basis of priority level. The CAM may be interrogated by line number in order to clear the waiting interrupt requests generated by a particular line. It is possible to get multiple matches in this situation if the priority of the specified line has been recently changed. However, the multiple matches are of no consequence since they are all simply cleared. When the CAM F bits are associatively tested to locate a vacant CAM word, the choice among multiple candidates is made by logic which selects the responding location having the highest conventional address.

When the CAM is interrogated on the juxtaposed E and level fields, what is sought is not an exact match as in other cases, but rather the word having the largest numerical value in those fields. This process is accomplished in the fashion described in Reference 6.

### INTERRUPT PROCESSOR OPERATION

IP operation can be divided into three independent phases: auxiliary functions such as changing the scratchpad contents (described earlier) or arming and enabling interrupts; inputting interrupts to the queued CAM; and outputting service requests to the CP.

#### Arming and enabling interrupts

Interrupts are armed and enabled by line number rather than level since priority levels are program-assigned. Interrupt lines may be armed, disarmed, enabled, and disabled both unitarily and in groups.

Unitary arming and disarming is accomplished with the arm register in the priority logic. The arm register bits are individually addressable and may be altered by transmitting one or more words from main memory.

The arm register can also be controlled in 8-line groups. Each 8-line group is represented by a single bit in a 8-bit (for the minimum 64-line configuration) to 32-bit (for the full 256 lines) control word. This allows all 32 8-line interrupt groups to be armed and/or disarmed in any combination with a single CP instruction execution.

Interrupts are enabled and disabled both unitarily and in groups through the scratchpad (and indirectly through the CAM). The scratchpad entry for each line includes an Enable (E) bit, so unitary enabling and disabling is performed by rewriting scratchpad words as described earlier.

Since the scratchpad is also sideways addressable in its LSB position, all 32 8-line interrupt groups can be enabled and/or disabled in any combination with a single CP instruction execution.

The Enable (E) bit in the CAM is updated whenever its counterpart in the scratchpad is altered. If a word is changed in the scratchpad, its line and level fields...
are used to search the CAM for a corresponding entry. If one is found, its E bit is modified to agree with the new value of the scratchpad E bit. In the case of a group operation on the combined scratchpad E bits, all those levels in the groups affected by the instruction would be subject to inspection.

Since the IP components function autonomously, facilities are available to provide program control over the activity of those lines whose parameters are being altered in the scratchpad and the arm register. The modifications (such as enable/disable, arm/disarm, priority reassignment, or a new pointer) are immediately reflected in any waiting requests. Alternatively these requests may be cleared, the modification taking effect only with subsequent new interrupts. The CP instructions which incur these changes include the capability to specify the following modes of operation:

a. Normal operation—the queued CAM continues to accept and process interrupt requests on all armed lines as usual.

b. Clear affected entries—each line changed in the scratchpad or arm register is looked up in the CAM, and if found the corresponding queue is completely cleared.

c. Hold all inputs—the queued CAM continues to process the waiting interrupt requests, but no new requests are accepted on any lines until all initiated changes have been completed. This is accomplished by inhibiting the priority logic response to interrupt signals.

d. Hold affected inputs—the queued CAM continues to process the waiting interrupt requests, but no new requests are accepted on any lines for which changes have been initiated but not completed. A comparator on the scratchpad address register causes those lines which fall between the current values of the scratchpad upper and lower limit registers to be considered disarmed.

e. Hold all outputs—the queued CAM continues to accept new interrupt requests, but no further CP service notifications are made for any lines until all initiated changes have been completed.

f. Hold affected outputs—the queued CAM continues to accept new interrupt requests, but further CP service notifications are made for lines for which changes have been initiated but not completed. The scratchpad address register comparator inhibits the IP from signaling the CP in response to interrupt requests on those lines which fall between the current values of the scratchpad upper and lower limit registers.

Any request which is inhibited this way also blocks all lower priority requests.

These CP instruction options are coded to allow combinations of input/output modes. The "hold" modes are applicable principally to cases where the scratchpad is being loaded from an image in main memory, or where a group enable in the scratchpad causes several CAM updates.

Detecting and acknowledging interrupts

When an interrupt signal is detected, it is loaded into the appropriate bit of the holding register. If the line is armed, it is gated to the priority tree where the highest priority line is encoded into an 8-bit scratchpad address. The source ID is stored in the queued CAM input data register, an acknowledge signal is returned to the device, and the line is disconnected from the priority tree until the interrupt is reset. Since the scratchpad gives precedence to the queued CAM, the priority logic may have to wait one cycle for service. When access is granted, the program-assigned priority level is retrieved and included with the source ID and line number in the queued CAM input data register.

Storing and scheduling interrupts

The word in the input data register is entered into the queued CAM by first associatively checking the level of the new interrupt request against those already there. If a match is found, this level is already assigned to a word in the CAM, and the new request is loaded into the rear of the corresponding queue. If there is no match, the request is placed in a vacant CAM slot. The CAM is then associatively searched for maximum on the combined E and level fields. If the entry retrieved is enabled but inactive (i.e., E set and R reset), the CP is trapped to initiate the new routine. Otherwise, no further action is necessary.

Any time that data entry is attempted and the CAM or queue is filled, a high priority executive trap occurs in the CP. The supervisory software then may arrange to simulate a portion of the queued CAM in main memory to handle the overflow at reduced rates, or it may decide to reduce the I/O traffic to within the queued CAM’s hardware capabilities.
Initiating CP service

There are three conditions under which the IP notifies the CP to initiate the processing of an interrupt service routine. The first occurs when a new interrupt request being loaded into the CAM is both enabled and higher priority than the currently active request. The second case is when the CP enables a level which is higher priority than the currently active level and which has a request pending in the CAM. Last is the completion of a higher priority interrupt routine when a lower priority request is enabled and waiting.

When one of these events occurs, the relevant line number from the queued CAM is put in the scratchpad address register. When access is granted, the IP signals the CP, sending it the priority level, source ID, and service routine pointer. These parameters presume a software supervisor in the CP, but are readily extended to interface with hardware-resident schedulers.

Terminating CP service

When execution of an interrupt service routine is completed, the CP returns the priority level of that routine to the IP. The corresponding CAM entry is deleted by resetting its F bit, allowing any pending request in the queue behind it to move into the vacated position. The E bit of the new request is updated from the scratchpad, and a new associative search for the highest priority waiting request is then initiated as described earlier.

Interrupt processor implementation

The IP consists primarily of memories which are inherently regular and thus readily lend themselves to economical batch fabrication. Implementation is entirely feasible with the level of integration available today in TTL, and will be further facilitated by impending improvements in semiconductor technology.

An MSI 8-input priority encoder circuit simplifies the task of detecting and recognizing interrupts. Three levels of these encoders are cascaded to attain the maximum 256-line fan-in. A high speed multiplexing scheme could be employed instead of the priority logic since both approaches establish a somewhat arbitrary order in which interrupts are accepted and stored. However, cost/performance tradeoffs favor the hierarchical technique with the logic functions currently available.

The scratchpad is conveniently constructed from internally decoded 64-word by 4-bit random access memory modules. The associated address, data, and limit registers, as well as the comparator, are also composed of standard MSI circuits.

A substantial reduction in hardware was achieved by adapting the CAM to take advantage of an existing associative memory element. Six of these 4-word by 4-bit blocks are needed for every 4-word by 23-bit CAM increment, plus some auxiliary logic to tie them together. The auxiliary logic would be included on the chip if a custom LSI circuit were fabricated for this application.

The 8-word by 21-bit queue behind each CAM position comprises sixteen 10-bit register packages plus a more general 8-bit register for the F bits. Again, the extra logic needed could be integrated into a special queue module.

Control and interface logic constitutes the remainder of the IP, or about 10 percent of the total hardware.

A maximum 256-line configuration having a 32-word CAM with 8-word queues consists of under 1500 IC's. Maintaining the same relative sizes of the CAM and the queues, a minimum 64-line IP would include only about 500 IC's. This suggests that even an expanded IP could be purchased for less than the price of a conventional I/O channel on many computers. The probable impact of eventual LSI implementation would be to reduce the chip count by at least an order of magnitude.

INTERRUPT PROCESSOR PERFORMANCE

There are four criteria commonly used to judge the performance of an interrupt system: reaction time, overhead, optimum priority response, and saturation.

Reaction time is the time between the occurrence of an external interrupt signal and the commencement of the first useful instruction in response to that signal. (This interrupt is understood to require that the CP pursue a higher priority task than is presently under way.) A maximum of 5.0 microseconds elapse in a 256-line IP from the moment an interrupt occurs until the CP is notified. This period is nominally 2.5 microseconds but the higher figure arises from worst case synchronization of the priority logic. Subsequent interrupts can be accepted by the IP every 1.4 microseconds and can also be sent to the CP at the same speed. The remainder of the computer's reaction time will be contributed by status preservation activities in the CP. A contemporary machine having a multi-level priority structure may be able to alert the CP this quickly, but it cannot so easily divorce itself from the CP's participation. When the liabilities of conventional interrupt hardware are compensated for in
the software, the supervisory bookkeeping can far outweigh the hardware delays.

Similar considerations apply to the interrupt completion procedure which is the second half of the interrupt system overhead. This overhead is defined as the difference between the time needed to completely process the incoming request and the execution time of all useful instructions. No more than 1.4 microseconds transpire when a 256-line IP terminates an interrupt, for a total IP overhead of 2.8 microseconds per complete interrupt cycle. Thus the IP can support a sustained throughput rate of almost 400,000 interrupts per second. Additional overhead factors in the machine are diminished by the power of the IP, but will be determined by the integrated hardware and software design of the CP's executive structure.

The independence of the IP's input and output functions implies occasional conflicts in accessing the scratchpad. These are resolved in favor of the queued CAM but the priority logic will not have to wait more than one scratchpad cycle since the CAM cannot supply consecutive CP service requests at the scratchpad cycle rate. In the worst case only one scratchpad cycle time (about 50 nanoseconds) is added to the IP input time.

The priority logic and queued CAM must also contend with occasional scratchpad updates by the CP. This is the lowest priority scratchpad function unless the acting CP instruction specifies one of the hold modes during updating. However, some interference may be precipitated by examination/modification of the CAM E bits as a result of scratchpad E bit changes. This alteration occurs in one CAM cycle time (about 100 nanoseconds) per affected entry.

Optimum priority response is a measure of the extent to which the computer is always executing the most important task as determined by the environment. The utilization of an autonomous IP for centralized control of interrupts assures that the other system processors are always devoted to the highest priority jobs without diverting their efforts to evaluate and manipulate interrupts.

To maintain accuracy in the priority scheduling, it is necessary that the lines which require very fast reaction time be attached to higher positions in the priority tree. Once in the queued CAM, all interrupts are served on the basis of their program-assigned priorities.

Saturation occurs when the system cannot respond quickly enough to all of the interrupts causing some of them to be disregarded. Protection against this is inherently supplied by the queuing that occurs in the IP.

CONCLUSION

A unique Interrupt Processor has been described which uses hardware to perform many of the interrupt handling functions heretofore dealt with by software in large on-line multiuser systems. The operation of this unit has been described and a brief look at its implementation and performance has been given.

The most significant aspect of the IP is the queued content-addressable memory which provides an efficient interrupt organization and storage facility. This queued CAM concept should also prove to be highly effective in many other hardware solutions to supervisory system problems normally handled by software.

REFERENCES

1. S ROSEN
   Hardware design reflecting software requirements
   Proc FJCC 1968 pp 1443-1449
2. E R BORGERS
   Characteristics of priority interrupts
   Datamation June 1965 pp 31-34
3. R J GOUNTANIS and N L VISS
   A method of processor selection for interrupt handling in a multiprocessor system
   Proc IEEE December 1966 pp 1812-1819
4. R J GOUNTANIS and N L VISS
   Methods of interrupt routing in a multiprocessor system
   IEEE Twin Cities Section November 1967
5. B W LAMPSON
   A scheduling philosophy for multiprocessing systems
   CACM May 1968 pp 347-360
6. M H LEWIN
   Retrieval of ordered lists from a content-addressed memory
   RCA Journal June 1962 pp 215-229
7. J G BENNET
   Letter
   Datamation October 1965 p 13
8. R V BOCK
   An interrupt control for the B5000 data processor system
   Proc FJCC 1963 pp 229-241