METHOD OF PREFETCHING DATA/INSTRUCTIONS RELATED TO EXTERNALLY TRIGGERED EVENTS

Technical field

The present invention relates generally to systems wherein an external source such as a scheduler in a network processor can interrupt the processor for processing a task wherein data/instructions are not related to the previous task and relates particularly to a method of prefetching data/instructions related to externally triggered events.

Background

The effectiveness of modern microprocessors and microprocessor cores strongly depends on the effectivity of caches since the instruction cycle time is much smaller than the memory access time. The caches exploit the locality of the memory accesses, that is the fact that a memory access is more probable to be near previous accesses.

The caches contain a mechanism, the cache controller, for loading selected areas (cache lines) with new contents, and in order to do so make room for this action by discarding old entries. The cache controller can currently be activated by software with a catch prefetch instruction (e.g. a Data Cache Block Touch for all PowerPC compliant devices). Furthermore, there are proposals for cache controllers to recognize regular access patterns like linear strides or linked data structures. Unfortunately, the present methods do not cover externally triggered events wherein the memory

contents needed in these situations are unrelated to the previous processing. In such a case, the only knowledge about the needed memory contents is the source of the event like interrupt sources, schedulers or other processors which assign the task.

In a system wherein an external source such as a scheduler in a network processor can interrupt the processor for processing data which are unrelated to the previous processed data, the processor generates a cache miss. This means that the processor stops processing until the data it needs is loaded from the memory into the cache. This wastes a considerable amount of time. Thus, with current memory technology and a processor clock speed of 400 MHz, every cache miss relates to 36 processor clock cycles, that means about 40 instructions. As the current technology trends show a stronger increase of the processor instruction rate than the memory latency, there is an increase of the number of lost instructions per cache miss.

Summary of the invention

Accordingly, the main object of the invention is to achieve a method of prefetching data/instructions related to externally triggered events in order to avoid the cache misses on data for which addresses can be determined easily.

The invention relates therefore to a method of prefetching data/instructions related to externally triggered events in a system including an infrastructure having an input interface for receiving data/instructions to be handled by the infrastructure and an output interface for transmitting data after they have been handled, a memory for storing data/instructions when they are received by the input

interface, a processor for processing at least some of the data/instructions, the processor having a cache wherein the data/instructions are stored before being processed, and an external source for assigning sequential tasks to the processor. The method comprises the following steps which are performed while the processor is performing a previous task: determining the locations in memory of data/instructions to be processed by the processor, indicating to the cache the addresses of these memory locations, Fetching the contents of the memory locations and writing them into the cache, and assigning the task of processing the data/instructions to the processor.

Brief description of the drawings

The above and other objects, features and advantages of the invention will be better understood by reading the following more particular description of the invention in conjunction with the accompanying drawings wherein :

- Fig. 1 is a block-diagram of a network processing system wherein the method according to the invention is implemented.
- Fig. 2 is a flow chart representing the steps of the method according to the invention.

Detailed description of the invention

Such a system includes a processor core 10 such as a PowerPC processor core equipped with a data/instructions cache. The system is structured by a high-performance bus 12 such as a Processor Local Bus (PLB) which provides the connection to an external memory 14 (e.g. A SDRAM) containing data as well as instructions by the intermediary

of a memory controller 16 providing independence of the bus architecture from the memory by generating for instance all necessary timing, refresh signals etc.

Bus 12 and memory 14 are also used by an infrastructure 18 which handles the data packets which are received from the network on an input interface 20. The infrastructure 18 manages the reception and transmission including assembly of packets, memory allocation and deallocation as well as insertion and deletion from packet queues.

Some of the packets do not require to be processed and are transmitted directly over the network by an output interface 22. Other packets have to be processed by processor 10. The decision whether a packet needs to be processed and which kind of processing has to be performed is done by a look-up and classify unit 24. In order to process a data packet, processor 10 needs several information. For this, it has to access the header of the packet and the additional information generated in the infrastructure 18. For instance, the infrastructure might have several ports where packets can arrive and the processor needs the information where the packet comes from.

A scheduler 26 handles all the data packets which need processing by the processor in one or several queues. These queues do not need to be physically present in the scheduler. At least the front entry of each queue has to be stored on-chip. This scheduler keeps track of the processor activities. When the processor has finished processing a packet, it requests the scheduler for the new task. But, if several queues with different priorities are managed, the scheduler 26 could also interrupt the processing by the processor of a low priority task in favor of a higher priority one.

In any case, the scheduler 26 has knowledge about the next task the processor 10 is going to process. The selected task determines which data will be accessed. In the case of a network processor as described here, the relation between the task (the entry of the queue) and the addresses first accessed, namely the packet header and the additional information is very simple. This translation from a queue entry into a set of addresses is done by an address calculation unit.

When the processor 10 processes a new packet and accesses data such as the packet header, it normally generates a cache miss if the invention is not used. This means that the processor would stop processing until the needed data is loaded from external memory 14 into the cache, which wastes a considerable amount of time as already mentioned. The cache prefetching according to the invention avoids the cache misses on data access which are sure to occur and whose addresses can be determined easily.

To work, the cache must be instructed to load the required data before the access happens. The initiation of this action comes from the scheduler which uses a direct connection 28 to the cache. After having determined the locations in memory wherein the packet header and the additional information have been stored, the scheduler assigns the addresses to fetch to the cache and the cache controller fetches the data from memory into the cache. After completion of this writing, either the scheduler interrupts the processor and assigns the new packet for processing when the new task has a higher priority than the previous task, or the scheduler waits for the completion of the previous task before delivering the new packet.

The method according to the invention is represented by the flow chart illustrated in Fig. 2. First, the infrastructure waits for the reception of new data, that is a data packet in the described example (step 30). The header of the packet is used for classification and the header and additional information resulting from the processing by the look-up and classify unit are stored into the external memory (step 32). The look-up and classify unit determines whether the packet needs to be processed by the software and determines its priority (step 34). If the packet does not need processing, the process loops back to waiting for receiving new data (step 30).

When the data packet requires processing, the scheduler has to calculate the address in memory which corresponds to the data the processor will access. In the example, it is the address of the packet header and the address of the additional information such as classifier result, input port (step 36). These addresses are then transferred to the data cache controller of the processor (step 38). The data cache controller writes the corresponding data into the data cache (step 40). This is done by interleaving the memory accesses created by the current packet processing.

At this stage, the process depends on whether the packet which has just arrived is of higher priority than the previous one (step 42). If so, the scheduler interrupts the previous task currently performed by the processor (step 44) and assigns the new packet for processing, and the processor starts processing and finds relevant data in the cache (step 46). If the new packet has no higher priority than the previous one, the processor has to complete the previous processing (step 48) before processing the new one (step 46).

Note that, in case of a packet having a higher priority, the scheduler has to wait for the completion of the data cache fetch before interrupting the processor. For this, the scheduler could observe actions on the bus and wait until all assigned accesses have been completed. Alternatively, the scheduler could wait a fixed amount of time or a direct feedback from the cache controller to the scheduler could be used.

It must be also noted that for two packets wherein the processing of the first packet is interrupted in order to process a second higher priority packet as above should happen in both cases on disjoint sections of the cache. Otherwise, the prefetched data could be erased from the cache before it is accessed. This can be achieved by using the mapping from virtual addresses to real addresses in the processor since the cache is usually indexed using virtual addresses.

Although the method of the invention has been described in a network processor environment, it will be clear for the man skilled in the art that it could be used in any system wherein the access to some data by the processor is sure to occur and whose addresses can be determined easily. In all cases, an external event is connected with some data to be processed. Thus, assuming that a new picture arrives at regular time intervals in a robot which uses a camera for navigation. The arrival of the picture is the event and the picture data itself is the associated data which is prefetched.

It must be noted that, for standard microprocessors, the address bus can be used as external source, as it is already observed for the cache coherency. In this case only one external wire is needed to indicate the prefetch request.

CLAIMS

1. Method of prefetching data/instructions related to externally triggered events in a system including an infrastructure (18) having an input interface (20) for receiving data/instructions to be handled by the infrastructure and an output interface (22) for transmitting data after they have been handled, a memory (14) for storing data/instructions when they are received by said input interface, a processor (10) for processing at least some of said data/instructions, said processor having a cache wherein the data/instructions are stored before being processed, and an external source (26) for assigning sequential tasks to said processor;

   said method being characterized in that it comprises the following steps which are performed while said processor is performing a previous task,

   - determining the location in said memory of data/instructions to be processed by said processor,
   - indicating to said cache the addresses of said memory locations,
   - Fetching the contents of said memory locations and writing them into said cache, and
   - assigning the task of processing said data/instructions to said processor.

2. Method according to claim 1, wherein said processor (10) is a network processor and the data to be processed are in the header of a data packet received by said infrastructure (18).

3. Method according to claim 2, wherein said external source is a scheduler (26) directly connected to said cache in said

processor (10), said scheduler determining the location in said memory (14) of data/instructions to be processed and directly indicating said addresses to said cache.

4. Method according to claim 3, wherein said scheduler (26) determines the locations of said data/instructions in said memory (14) by calculating said addresses.

5. Method according to anyone of claims 2 to 4, wherein the step of assigning the task of processing said prefetched data/instructions consists in interrupting the processing of a previous packet and starting the processing of a new packet which has a higher priority than said previous packet.

6. Method according to any one of claims 3 to 5, wherein said cache is associated with a cache controller which is in charge of fetching the contents of said memory locations whose addresses have been determined by said scheduler (26) and writing them into said cache.

7. Method according to claim 5, wherein a Processor Local Bus (PLB) is used by said processor (10) and by said scheduler (26), said scheduler interrupting said processor after determining the completion of the data cache fetch by monitoring said bus and observing precisely when the data returns from said memory (14).

8. A system comprising means adapted to achieve the steps of the method according to claims 1 to 7.