INDIAN INSTITUTE OF TECHNOLOGY
DELHI

COL216 - ASSIGNMENT 5

by

Hardeep Kaur Anjali Sharma
2019CS10354 2019CS50422

Professor:
P. R. PANDA

17 May, 2021

DRAM Request Manager

APPROACH

To extend the DRAM request manager to multi core scenario, an interface between various cores and DRAM is provided by Memory Request Manager.
AIM : To increase throughput (total number of instructions completed by the whole system in a given period)
We measure the throughput by CPI (Cycles Per Instruction)
Following are the key features in our approach :
1. Remove the redundant instructions
2. Forwarding
3. Each core uses relative memory addresses so as to avoid disturbing the semantics of programs.

DESIGN

Following are the key features in our design :
1. Multilevel Cache, with a private cache for each core
2. Finite Size of waiting buffer
3. Prefetching of instructions, to increase throughput
4. Incorporation of MRM delay to the working of system
5. Forwarding of (data transfer) instructions when possible

Memory Request Manager

When any core encounters a data transfer instruction,it sends a DRAM request,which goes to MRM(Memory request Manager)

CASE 1

If there is no ongoing instruction ,DRAM request goes to the DRAM directly.This DRAM request owuld execute in next cycle.Thus,the delay involved is of sending the request from MRM to DRAM is 1 cycle.

CASE 2

If there is an ongoing instruction,DRAM request is stored in the buffer in that cycle itself.The instructions are stored in the buffer by implementing a linked list,which stores the distinct row numbers which to access in DRAM.

Thus whenever there are more then one instructions in buffer corresponding to the same row,it gets added to the linked list of respective row,thus helps in easier memory reordering.

WHY IS MAXIMUM NUMBER OF CORES 32?

When looking at raw performance alone (total number of instructions executed within a unit of time), more but less powerful cores clearly outperform chips with few but powerful cores, within a set power budget. This is a tempting prospect, especially in domains with abundant parallelism such as networking, graphics, web servers etc. Accordingly, this category of chips—with many, but simpler cores—is usually represented by processors targeting a specific domain.In this chapter we survey some of the best known representatives of this category:- Tilera's Tile GX family, NVIDIA's Graphics Processing Units (GPUs), pi-coChip's 200-core DSP as well as Intel's recently announced Many Integrated Core (MIC) architecture.

Mutliple Instructions

If there are multiple instructions in the same cycle for MRM,only one of them is set as current instructions and the others are stored into the buffer. The respective requests are in the respective cache memories.

Sending requests from MRM to DRAM

CASE 1

When there is no request going on in DRAM,the instruction in MRM goes to DRAM in same cycle.

CASE 2

If there is an ongoing request then requests are stored into MRM in 1 cycle.

Choosing next instruction

When there are pending instructions in the buffer,the instruction to be executed next is chosen to minimise number of cycles.In the penultimate cycle,MRM checks for any request with for the same row in the buffer (reordering).Although,the check happens after the current request is executed.

CASE 2.1

If there is a request for the same row in the buffer,then it is sent to DRAM in next cycle (saving the cycles used in row access).

DELAY : The delay in sending request from MRM to DRAM should be 1 cycle,but does not add since it happens when the previous instruction is executing.

CASE 2.2

If there is no such instruction,then we check if an instruction with same row comes in the ongoing cycle.Also,this check occurs simultaneously with the instructions being stored in buffer by MRM.

CASE 2.2.1

If such instruction is present,MRM fetches it in 1 cycle and send it to DRAM in the next cycle

DELAY: If the instructions was fetched in this cycle only,then it would have a delay of 2 cycles.

Whereas if the instruction was in the core's cache, it would take 1 extra cycle,thus the delay would be of 3 cycles.

In any case the delay would be of 2 or 3 instructions only,depending on whether it was in cache or not.

Although,net observed delay would still be of 1 cycle only,as the fetching and checking of instruction part is done when previous instruction is executing in DRAM.

MRM in Brief

Final writeback is performed if it can be done within the range M.

1. MRM idle no DRAM curr-req 1 cycle fetch instruction and send to DRAM

2. DRAM currreq is there

• 2.1 no other request is being sent to MRM, 1 cycle to push it to MRM buffer.

• 2.2 Some request is being sent in this cycle, store this instruction in cache, until MRM gets free. Then, push it to buffer later in 1 cycle.

• 2.3 In the cycle of column access,

◦ 2.3.1 check if any request of same row is there, if yes, send it to DRAM in 1 cycle.

◦ 2.3.2 If not, check if any request scheduled to come in this cycle has same row, if yes, prefetch it in this cycle and send it to DRAM in next cycle.

◦ 2.3.3 If no request of buffer row could be prefetched either, send the request lying next in buffer to DRAM in 1 cycle.

MRM is involved in 1 cycle : fetch and send 1 1 cycle : push to buffer 2.1 2.2 1 cycle : send to DRAM 2.3.1 2.3.2 2.3.3 1 cycle : prefetch 2.3.2

Buffer Size

The size of the buffer has been set to 64.Also there is no limit on size for any individual core.That means if there is only single core running then the buffer can store as much as 64 requests at a time.

Cache Memory

There is a cache memory provided to hold the DRAM requests of each core. The limit to this is also set as 64.

This has been done to fully utilise MRM buffer.If all requests are from single core,then MRM buffer would store all of them and they would also be available in the core's cache.

The buffer limit is set by contemplating the number of cores versus the number of data instructions needed in general programs. We have set the maximum number of cores as 32.Thus buffer size of 64 and a cache memory size of 64 gives us the most optimised result.

Since,the buffer size is limited,if there are more than 64 instructions at a time they would have to wait till some instructions from the buffer go to MRM.

When the instruction is stored in buffer,processor does not goes on executing those instructions,as that would be mean more than 64 instructions in buffer.Furthermore,if this happens there would be 64 instructions for each core including the buffer.

Each core has a private cache,which helps in reordering the instructions easily.Since,the DRAM request has to be independant of the previous instructions in order to reorder,which is easier to check for each core separately.This also helps in all the other optimisations ,i.e. lw-sw forwarding and removing redundant instructions as these have to check all the instructions in a single core only.

STRENGTHS

Features other than mentioned in design are as follows:

1.Redundant requests are removed, thus saving the cycles needed for their execution.

2.Reordering is done to minimise the number of clock cycles needed.The updates related to same rows are executed first,thus substantially decreasing the number of row access delays.

3.All the independent instructions do not have to wait to get executed(can happen in same clock cycles.

4. PREFETCHING is implemented.

WEAKNESSES

1. If the load/store instructions to be executed are dependant,the queue would be empty.In such a case there is no reordering done.All the instructions execute sequentially(as in Non blocking memory)

2.Since the memory is now divided into N parts,if the instructions try to access more memory than that it would raise an error

3. Since number of cores is usually less(always finite), the processor still has a sufficient amount of memory in any case.

TESTING STRATEGY

Command line should consist of number of CPU cores,simulation time,row access delay and column access delay values. A 1 could be written in command line to view the details of each cycle. If the command line is wrong, an error is printed saying the input is to be checked

If the input file is not available, unable to open file is printed. Since we can initialise registers and data memory .it would give a warning to the user,that these files were not initialised.

All the test cases are submitted along with the report contain a Readme describing which functionality is checked in that particular testcase.