Mechanism for Enhancing the Overall Performance of Multicore Processors

Pankaj Rakheja, Charu Rana, Mandeep Singh Narula

Abstract— Multicore architectures are focused on improving the performance of the processor however their performance depends on the thread level parallelism of the application program which is difficult to extract and the design and production of multicore architectures is through a unreliable fabrication technology which imposes significant barriers to lifelong reliable operation of chip as they are vulnerable to defects and disturbances. In this paper we are proposing a mechanism to enhance overall performance of the multicore processors by adopting multiple cache cores and check cores with improvement in software managed L1 cache of computation core and algorithm implemented there to access right cache core to reduce cache miss and memory access frequency and to isolate it at right instant to prevent degradation in performance in case of L2 cache miss in cache core. We have designed a mechanism which will try to eliminate the defects in redundant as well as non redundant logic structures in the core for enhancing its performance and efficiency. We are stressing on thread scheduling, thread swapping and core salvaging at micro architectural level that is at the basic gate levels in the core which enhance overall performance and efficiency of the processor which can be aided by Intel quickpath interconnect technology and frequency scheduling that can reduce power consumption, speed up as well as optimize the core to core communication.

Index Terms— Cache, checker, Thread

1 INTRODUCTION

A multi-core processor is an integrated circuit (IC) in which we have two or more processors for enhancing the performance, reducing power consumption and having more efficient simultaneous processing of multiple tasks. The composition and balance of the cores in the multi-core architecture shows great variety. Some architecture employs only one core Design repeated consistently which are known to have homogeneous cores, while others use a mixture of different cores, each optimized for a different role they are known to have heterogeneous cores. The general trend in processor development has moved from dual-, tri-, quad-, hexa-, octo-core chips to ones with tens or even hundreds of cores. This has been made possible through advancement in semiconductor technology and it is expected that number of cores will increase. These advances have sustained the validity of Moore's law for several decades both in device count and performance.

The thread level parallelism can be executed efficiently in multicore processors however this technique needs more parallelism in programs with increase in number of cores. If we are unable to exploit the thread level parallelism then performance gain will deteriorate.

In today's scenario cost effective dependability for general purpose computing is demanded unlike before where high dependability/reliability was necessary for few applications and systems only. Major challenge to this is the unreliable technologies employed for manufacturing multicore processors and memories.

Some effective hardware and software solutions will have to be developed for increasing dependability on multicore chips and computing systems which are built via unreliable techniques.

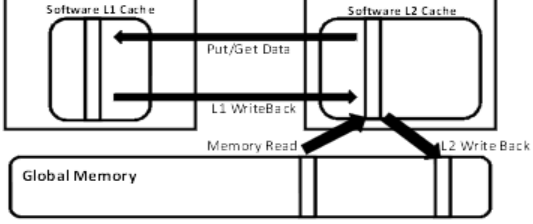
In this paper we are proposing a mechanism considering both these aspects of the scenario that is with extracting and exploiting thread level parallelism in multicore architecture through thread scheduling and thread swapping algorithms, we are compensating various flaws which may creep in due to unreliable techniques employed for developing these structures by adopting core salvaging and faulty component isolation at micro architectural level.

- *Pankaj Rakheja is Assistant Professr in Institute of technology and Management,Gurgaon,India,. E-mail: pankajrakheja@itmindia.edu*
- *Charu Rana is Assistant Professr in Institute of technology and Management,Gurgaon,India,. E-mail: charurana@itmindia.edu*
- *Mandeep singh Narula is Assistant Professr in Institute of technology and Management,Gurgaon,India,. E-mail: msnarula@itmindia.edu*

2 OVERVIW

A. Enhancing performance through thread level parallelism

By employing parallelism in a given program multicore processor can be executed efficiently. So we have to extract thread level parallelism in program which is quite difficult but mandatory to have good performance gain. For that we can use idle excess cores on the chip. Yosuke MORI and kenji KENSE have proposed a cache core mechanism[1] where the excess core behaves like an L2 data cache managed through a software program. They have utilized the fact that the communication overheads in chip multiprocessor between on chip cores are smaller than between core and off chip memory. And global memory access frequency will also reduce. Here if the program has to access the data in the global memory it first accesses the L1 cache. If L1 cache miss occurs then it accesses the L2 cache of the cache core. They have used one excess core as cache core we can opt for multiple cache cores which will be accessed based on the instruction or application program running in the main computation core to improve the hit ratio of the L2 cache in cache core. And we need to design a mechanism that cache core is isolated at the right time to improve degradation of performance by the L2 cache miss. Figure 1 below shows communication amongst computation core, cache



Moreover we need to make an improvement in the L1 cache employed at the computation core by managing it through an improved software code. For enabling all this we will use virtual hardware and prefetcher code.

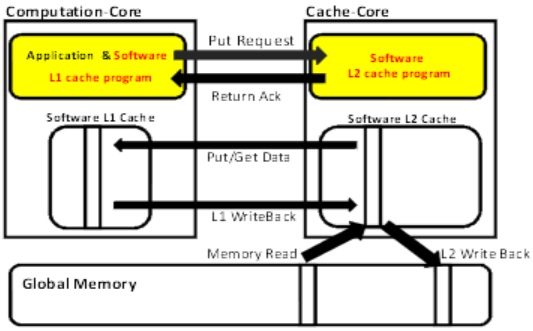


Figure 1. Cache core architecture

B. Compensating flaws due to unreliable techniques

With advancement in semiconductor technology number of cores on chip has increased and performance has improved too. But fabrication technologies adopted are not that reliable because of which some errors or defects creep in the system. The most important sources of unreliable hardware operation can lead to system failure are process variability that may lead to heterogeneous operation of identical components, transient errors in submicron circuits and ageing effects due to extreme operating conditions. To compensate these we can use online error detection, recovery and repair schemes that can guarantee low cost dependability. For error detection we may employ these approaches redundant execution, periodic built-in self test, dynamic verification and anomaly detection techniques. Where in redundant execution two independent threads execute copies of the same program and results are compared; in periodic built-in test perform non coherent error detection by executing periodic self tests; in dynamic verification we employ dedicated hardware checkers to verify validity of specific invariants and these are carried out at run time and in anomaly detection we monitor anomalous behavior or symptom of faults.

For error recovery and repair we can opt for two basic recovery techniques: Forward error recovery (does not require roll back to previous correct state) and backward error recovery (requires roll back to previous correct state). The redundant and non essential components can be disabled to improve yield and performance. Repair techniques rely on the coordination of the fault diagnosis and isolation at different levels i.e circuit or architectural level.

Software anomaly treatment method [2] proposed by Pradeep ramachandran, Siva kumar sastry hari, sarita V. adve is a comprehensive solution to detect, diagnos and recover from variety of hardware faults by observing anomalous behavior of system requiring fault detection, fault diagnosis and fault recovery components.

Dynamic verification of cores and memory system proposed by Daniel J.Sorin, Albert meixner is to check the certain system wide invariants rather checking specific components. It is independent of implementation and can detect errors due to soft and hard faults.

Accidental heterogeneity can be dealt by core salvaging proposed by Arjit Biswas that allows faulty core to continue operation. It avoids usage of faulty part of the core rather than disabling whole core.

Here in this paper we will concentrate more on non redundant logic employed in many redundant structures like multientry arrays made from decoders, buffers along with interconnects. We will also lay stress on improving thread scheduling and thread swapping algorithms which can be employed in multicore processors to enhance their performance.

C. Per-Core Frequency Scheduling

Multicore architectures offer a potential opportunity for energy conservation by allowing cores to operate at lower frequencies. Existing analytical models for power consumption of multicores assume that all cores operate at the same frequency where off-chip voltage regulator used to set all sibling cores to the same voltage level [3]. For off-chip regulators, cores on the same chip must operate at the same frequency and in case of multiple chips, cores on different chips may operate at different frequencies, [4]. With the help of Turbo Boost [5] technology, better performance can be achieved by boosting all cores to a higher frequency, only when the processor is operating below rated power, temperature, and current specification limits. Studies have shown that significant energy can be achieved by controlling each core voltage. Recent advances leads to on-chip multicore voltage regulator (MCVR) which can accept an input voltage and scale it down to a range of voltages to cut power according to CPU demands. [6]

A fine grained model developed [7] after exploiting these technologies for energy efficient computations and management of resources. An energy aware resource management model is utilized in this paper to reduce the power consumption by multi cores operating at different frequencies and to provide a mechanism for creating schedule of resource usage and frequencies at which processor cores should execute to complete computation.

D. Intel QuickPath Interconnect Technology

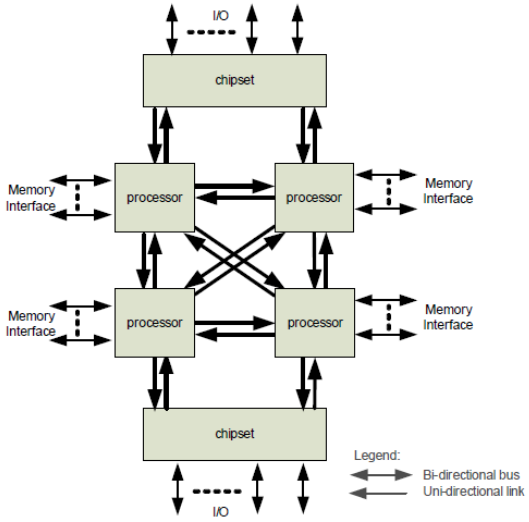


Figure 2. Intel quickpath interconnect

Intel Quickpath interconnect technology [8] shown in figure 2 provides high speed, point to point connections between microprocessors and external memory, and between microprocessors and the I/O hub. It is implemented in Intel's latest generation microarchitecture along with integrated memory controllers and distributed shared memory architecture.

In this technology each processor has its own dedicated memory which can be accessed through memory controllers. The dedicated memory of another processor can also be accessed using high speed Intel Quickpath Interconnect which links all processors. This technology provides high bandwidth, low latency and enhances application performance and reliability for a wide range of multi-core systems.

3 MECHANISM DESIGNED

We are here concentrating on all the aspects affecting performance of the processor. The mechanism proposed here will try to optimize the performance with marginal tradeoff between speed and efficiency of the processor. After analyzing the whole scenario the areas which need to be improved are L1 cache; processor to processor communication; Interconnections; errors crept in due to non reliable fabrication technologies; errors due to malfunctioning of non redundant logic; processor to memory interaction; Clock frequency allotment and obtaining parallelism in application program. While solving these issues we have take into consideration speed, accuracy, cost and efficiency of processor they all need to be optimized. For that work has to be done in every sphere of operation of processor at both hardware and software level.

A.

Improvement at Hardware level

Here we have to improve the existing hardware and fabricate new one on chip if necessary. We have multiple cores on single chip out of which certain are idle. We can utilize these cores which will in turn reduce need of extra hardware. These cores can be used as cache cores, checker core and communicator core. Cache core will act as a data cache for the main computation core it will just behave as L2 data cache for the computation core. Checker core will be equipped with a software program that will be capable of seeking erroneous non redundant logic in the other cores and replacing that erroneous hardware by a software module giving the same result. So here we will be salvaging at micro architectural level rather at architectural level. While the communication core will act as controller for computation core to cache cores communication it will be equipped with a software program which would give it a self learning capability just like routers in the networking so it will intelligent enough to access right cache core for seeking data to reduce hit ratio. And moreover the interconnections between these cores can be deployed through Intel's quick path interconnect which will speed up the whole process the type of interconnect deployed will be decided by the rate at which data need to be transferred that is on the basis of application. And levels of cache need to be aggravated size need not be increased much but the division of cache on the basis of the type of instruction it would be storing that is dependable and undependable execution instruction will further enhance the processor performance as hit ratio will improve this need to be managed through software. Frequency allocation for clock should be done on demand basis the idle processor should be operated at low frequency to reduce power requirement.

B.

Improvement at Software level

Improvement at software level is more important as we reduce extra hardware requirement which will reduce cost. First the improvement needs to be done at L1 cache in computation core it should managed properly division of cache into multiple levels can be done through software managed program which will store frequent instruction opcodes on basis of the their type and operation they perform which will reduce the seeking time and thus hit ratio will improve too. This can be done by observing or monitoring control signal status during fetching cycle. Then if data cache miss is found then the cache core can be accessed. Multiple cache cores can be used so that if one is busy then other can be referred. Then the communication amongst them can be improved through Intel's quick path interconnect. Software instruction opcode fetcher can be implemented and the corresponding control signals generated can be buffered for each instruction.

The overall processor with multiple cores with these improvements can be seen as below in figure 3 which shows conceptual overview of the whole scenario

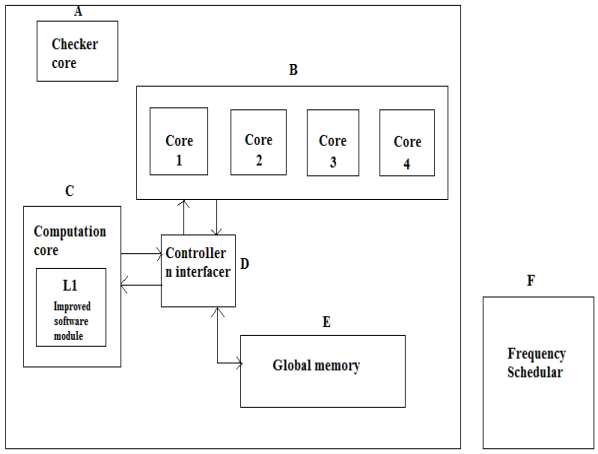


Figure 3: Conceptual overview

A. Checker core

It will be equipped with a software module which will look for any discrepancies in the functioning of the core and will identify the malfunctioning of core by dynamic verification method which analysis certain parameters on which performance of the core depends. Basic algorithm is shown below

Step1: Depending upon the core architecture n application determine parameters for analyzing performance

Step 2: Then do periodic analysis of these parameters to look for any discrepancy
Step3: If any discrepancy found then execute the corresponding module
Step4: go back to step 2

B. Multiple cores

It comprises of multiple cores which are equipped with software that enable them to be used as cache core which will basically store data. Here an idle core can be used as a data cache. The advantage of having multiple data cache is that if the core being used as a data cache has some urgent work to do or has encountered an interrupt it could transfer its data to neighboring core through controller n interface in between so as the computation core has access to that data throughout. And moreover performance of the core is the effected. Basic algorithm is shown below

Step1: Cache core on encountering an interrupt moves to step 3
Step2: Execute L2 cache software module indefinitely
Step3: Transfer the data to neighbor cache up to down and to interface and controller down to up so that computation cache has access to data mean while
Step4: Execute ISR

C. Computation core

The core which does computation is called computation core it will be equipped with an improved L1 instruction cache along with L2 data cache. The software will generate L1 cache where instruction opcode will be stored as per its type in order to improve cache hit ratio and L2 data cache will also be maintained their itself to reduce core to core communication if possible. Here an instruction prefetcher can be employed which prefetch all opcodes of program to increase parallelism. Basic algorithm is shown below

Step1: Divide the L1 cache into multiple levels
Step2: Store instruction opcodes in respective cache
Step3: Access the right cache depending on its type

D. Controller and Interfaces

It aids in computation to cache core communication so as to speed up the processor it will be equipped with a fully fledged software module, a processor to run that module and a small inbuilt memory.

Step1: Send control messages
Step2: Monitor core Cache access request
Step3: Give Direct cache core access to computation core and If interrupt comes on busy pin move to next step
Step4: Move data from cache to its temporary memory to give access to computation core n mean while transfer the data to new cache core
Step5: Move back to step2

E. Global Memory

Here it is the memory shared amongst all the cores which is accessed via the controller or through dedicated lines not shown in figure 3.

F. Frequency scheduler

It will assign frequency or clock signal to the cores on the basis of status of the core that is if needs to work at bulk of data at a time or needs to speed up higher clock will be given and vice versa. Basic algorithm is shown below

Step1: Determine the core demand
Step2: Switch its clock source
Step3: Goto step1

4 CONCLUSION

Multicore architectures are focused on improving the performance of the processor however their overall performance and speed depends on the thread level parallelism, technique of fabrication, fault detection and recovery, the type of interconnections deployed, cache maintenance algorithms etc. Here in this paper we tried to cover all aspects over which the performance depends and suggested requisite steps which need to be carried out in order to enhance the speed, reduce power consumption, utilizing the available resources at sake to full and to manage multiple cores in best possible way. The concept of multiple cache cores, controller and interface, checker core and Quick path interconnect will optimize the overall scenario. Future work comprises of implementing in on simulator to analyze by what factor performance improves and then burning it on chip and analyzing real time constraints.
.

5 REFERENCES

[1] Yosuke Mori, Kenji Kise, “The Cache-Core Architecture to Enhance the Memory Performance on Multi-Core Processors”, International Conference on Parallel and Distributed Computing, Applications and Technologies, 2009.

[2] Arijit Biswas et. al., “Architectures for Online Error Detection and Recovery in Multicore Processors”, EDAA, 2011

[3] Naveh, E. Rotem, A. Mendelson, S. Gochman, R. Chabukswar, K. Krishnan, and A. Kumar, "Power and Thermal Management in the Intel Core Duo processor," Intel Technology Journal, vol. 10, no. 2, pp. 109-122, 2006.

[4] X. Zhang, K. Shen, S. Dwarkadas, and R. Zhong, "An Evaluation of Per-Chip Nonuniform Frequency Scaling on Multicores," in Proc. of USENIXATC, 2010.

[5] "Intel Turbo Boost Technology in Intel Core Microarchitecture (Nehalem) Based Processors," White paper, Intel, November 2008.
[6] W. Kim, D. Brooks, and G.-Y. Wei, "A Fully-Integrated 3- Level DC/DC Converter for Nanosecond-Scale DV S with Fast Shunt Regulation," in Proc. of ISSCC, 2011.
[7] Xinghui Zhao and Nadeem lamali, Fine-Grained Per-Core Frequency Scheduling for Power Efficient Multicore Execution, Proceedings of the 2nd IEEE International Green Computing Conference (IGCC 2011, pp:1--8, July 2011.
[8] An Introduction to the Intel QuickPath Interconnect, White Paper,Intel, January,2009.