Scalable and Flexible heterogeneous multi-core system

Rashmi A Jain,
Electronics engineering department
G.H.Raisoni College of Engineering Nagpur
(M.S.) India

Dr. Dinesh V. Padole
Electronics engineering department
G.H.Raisoni College of Engineering
Nagpur (M.S.) India

***Abstract*—Multi-core system has wide utility in today's applications due to less power consumption and high performance. Many researchers are aiming at improving the performance of these systems by providing flexible multi-core architecture. Flexibility in the multi-core processors system provides high throughput for uniform parallel applications as well as high performance for more general work. This flexibility in the architecture can be achieved by scalable and changeable-size window micro architecture. It uses the concept of execution locality to provide large-window capabilities. Use of high memory-level parallelism (MLP) reduces the memory wall. Micro architecture contains a set of small and fast cache processors which execute high locality code. A network of small in-order memory engines use low locality code to improve performance by using instruction level parallelism (ILP). Dynamic heterogeneous multi-core architecture is capable of reconfiguring itself to fit application requirements. Study of different scalable and flexible architectures of heterogeneous multi-core system has been carried out and has been presented.**

Keywords-Flexible Heterogeneous Multi Core system (FMC); instruction level parallelism, thread-level parallelism; and memory-level parallelism; scalable; chip multiprocessors (CMP).

I. INTRODUCTION

A multi-core system is a single computing component. It has two or more processors which are independent of each other and each is called as a core. Each core reads and executes program instructions.

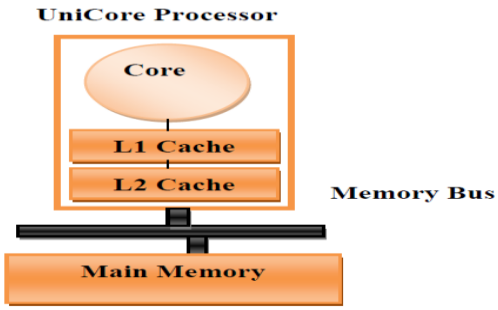

Fig. 1 Uni-core systems

The multiple cores can run multiple instructions at the same time that causes increase in overall speed of program execution. Multiple-core processors are also called as single-chip multiprocessors or more simply chip multiprocessors (CMP).

They appear similar to a traditional symmetric multi processor (SMP) but with all of its processors located on a single chip. In processors of today, there are typically 2 or 3 levels of on-chip cache

The level 1 (L1) cache is neighboring to the processor execution core and has the direct access time but the lowest capacity. Thelevel2 (L2) cache is next in the cache hierarchy and has longer access times but larger capacity. Finally, there may be a level3 (L3) cache with even longer access times but even larger capacity. On multi-core processors, the last-level cache, which is the level before requiring off-chip main memory access, is usually shared among more cores. Typically, this component is the L2 cache or the L3 cache.

Designer integrates the cores onto a single integrated circuit known as a chip multiprocessor or CMP. Processors were made with only one core. A many-core processor is also multi-core processor. In multi-core systems, the term multi-CPU refers to multiple physically separate processing-units. The many -core and multi-core are sometimes used multi-core architectures with a high number of cores tens or hundreds.

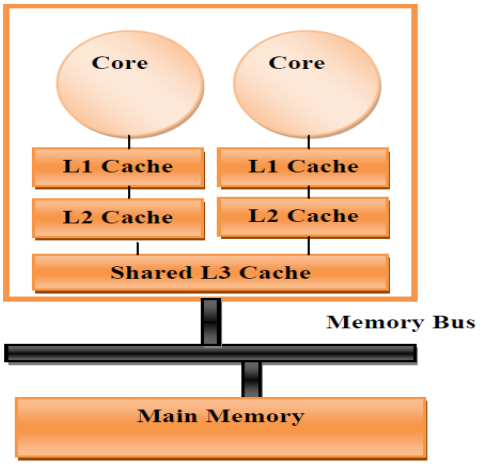

Fig. 2 Multi-core systems

A dual-core processor has twice cores a quad-core processor have four cores, it means it is made by four core a hexa-core processor have a six cores and similar like an octa-core processor have eight cores. We propose a micro architecture capable of running a single thread or many threads with high performance.

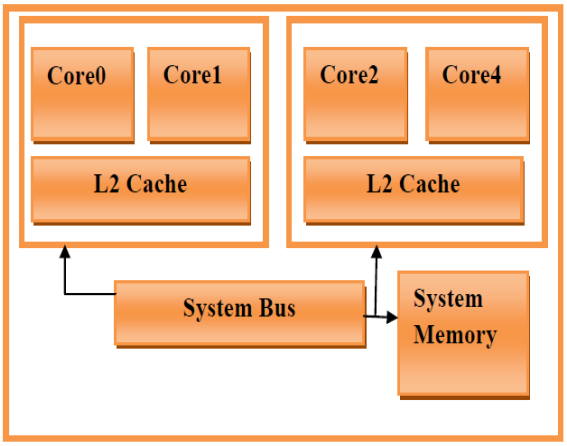

Fig.3 multi-core processor

It may couple cores in a multi-core device tightly or loosely. For example, cores may possibly or may not be share caches, and they may apply message passing or shared memory inter-core communication methods. General network topologies to interconnect cores consist of bus, ring, two-dimensional mesh, and crossbar.

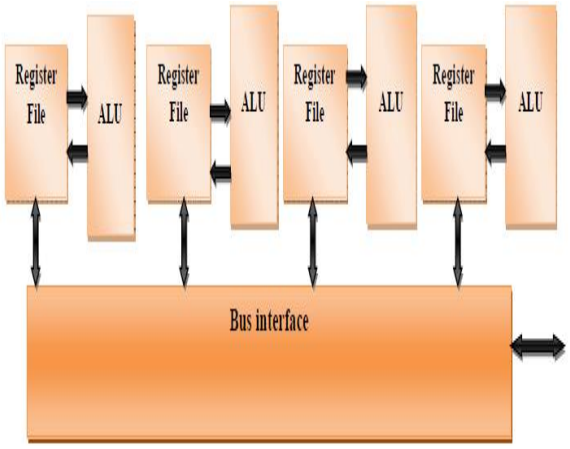

Fig. 4 Multi-core Architecture

A. Types of multi-core system

1. Homogeneous multi-core systems-Homogeneous multi-core systems have only identical cores. Some system use one core design repeated consistently known as homogeneous multi-core systems.
2. Heterogeneous multi-core systems-Heterogeneous multi-core systems have cores which are not identical. It means use a mixture of different cores known as heterogeneous multi-core systems.

B. There is two kinds of heterogeneous multi-core processor

1) Fixed heterogeneous multi-core processor-Fixed heterogeneous architecture in which partitioning remains static and it only roughly fits application requirements.

2) Flexible heterogeneous multi-core processor-There is dynamic heterogeneous multi-core architecture capable of reconfiguring itself and fit application requirement without programmer interference.

C. Advantages of multi-core system

1) Multitasking: -Each system has two processing cores for a maximum of twice the operating power and for better multitasking. Major advantages of multi-core system are heavy multitasking.

2) Application Support: - New applications are take advantage of this technology by using a technique known as Multithreading.

3) Power saving: - Multi-core systems have the ability to turn off one of their cores when application demand is low to save power.

Through the rest of the paper we will describe the micro architecture of our multi-core approach.

II. FLEXIBLE MICRO-ARCHITECTURE (FMC)

The basic of this architecture is a scalable, variable-size window micro-architecture that uses the concept of execution locality to provide large-window capabilities.

A). ILP (Instruction-level parallelism)-It calculate the many operations in a computer program can be performed simultaneously. B).TLP (Thread level parallelism) -It is a form of parallelization of computer code across multiple processors in parallel computing. c). MLP (Memory Level Parallelism) - In computer architecture the ability to have pending multiple memory operations. In a lone (one) processor, MLP may be measured a form of ILP, instruction level parallelism.

Micro-architecture uses ILP by having an effective instruction window of thousands of instructions spread across the processing elements, largely overcoming the unhelpful effects of long-latency memory operations. And it is also uses TLP for comparable workloads by allowing multiple threads to automatically allocate the processing elements. It needs to realize the greatest performance, quite than giving each thread the same kind of core of its needs. These advantages are obtained lacking changes to the ISA, compiler or operating system.

The fundamental property of the FMC is its ability to Change the instruction window size at runtime. It is able to do so by with dynamism adding or removing memory engines from the system. This property allows the processor to get used to the requirements of the application and activate only those Memory engines that are lead to improved performance.

Scalable multi-core architecture consists of a set of Cache Processors, every one with a static partition of memory engines, and a group of memory engines that can be dynamically assigned to the different threads. Figure 5 shows a general view of this micro architecture.

The architecture of FT64-3 processor contains one scalar core, one stream core, one 512KB shared secondary level cache and one DDR2 (direct device register) memory controller. Multiple processors can be connected by network interface directly. Scalar core can issue multiple instructions concurrently, and has two float pipelines, independent L1I-cache and L1Dcache, and also supports data consistency with other processors. Scalar core is responsible for running as, processing basic operations, scheduling instruction stream and

data streams for the stream core, and also managing communication with off-chip. The stream core, derived fromFT64-2processor, serves as an accelerator of scalar core. Its architecture is close to that Of Imagine stream processor , which contains 16 sets of double precision FPU (grouped as 4 cluster),µc(micro controller), Stream Register File (SRF), Stream Controller(SC) etc. It is responsible for executing stream instructions received from scalar core, and moves stream data on demand. Under the control of microcontroller, 4 clusters run by means of SIMD (single instruction multiple data). After stream data is loaded into SRF, SRF transmits stream data among clusters, µc etc, and writes results back to L2Cache.

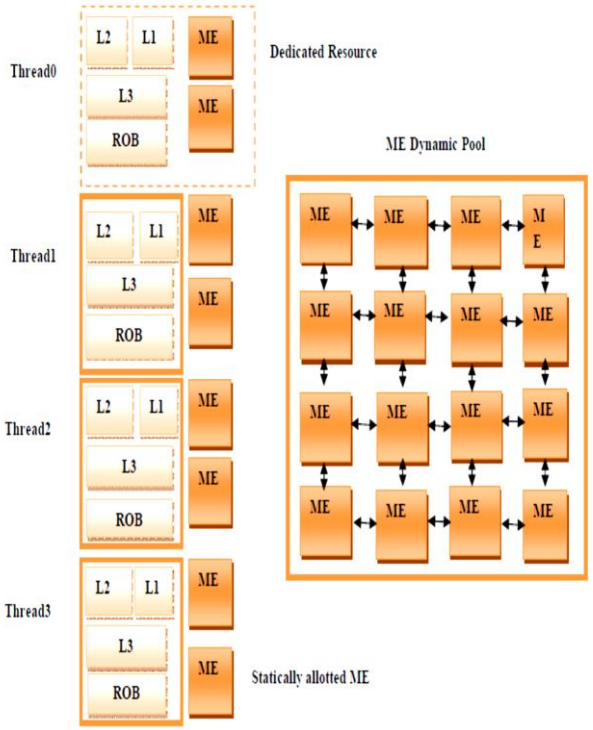

Fig.5 the micro-architecture of the flexible multi-core micro-architecture

Including a set of Cache Processors, 3 statically assigned ME (memory engine) per thread, and a dynamic pool of memory engines.

A. Memory architecture in processor core

The internal memory of scalar core is composed of common register file, 8KB Ll I-cache and 8KB Ll D-cache an it is similar to memory of traditional processor. It has execution model to uses a MLP (memory level parallelism) can tolerance latency sensitivity and bandwidth sensitivity.

III. Related Works

In addition, this design features only two execution modes that is 1.Small window or 2.Full window.

This makes it flexible chip multiprocessor. In this use the decoupled nature of this approach but overcome its limitations. Allow it to scale too many cores and many threads. The result is a processor with a variable window/issue size using a simple scalability mechanism.

Variable-size window processor .It uses multiple small cores, called memory engines. Linked through a system, to compute memory dependent instructions.

The network introduces (reduce the latency) latency, but this additional latency has little impact on instructions already waiting hundreds of cycles due to a cache miss. The memory engine network (different method for sharing the threads) can then be shared among threads to build a reconfigurable heterogeneous multi-core architecture. [1]

They proposed a new micro architecture that significantly improves performance by overcoming memory latencies while keeping complexity within reasonable limits. And they proposed a scalable micro architecture with a variable window size that can be tuned by adding or removing memory engines. They proposed a multi-threaded implementation of the micro architecture, the first heterogeneous multi-core architecture that adapts dynamically to the requirements of the threads.

Heterogeneous multi core processor can combine qualities of different architecture; it can reach peak performance as high as processors with unique architecture, though keeping as flexible as established general purpose processors at the same time. In this equivalent stream memory sub-system architecture for FT64-3 is presented. [2] It is proved that the LLC (last level cache) miss penalty is a better metric to achieve scheduling on heter-CMP system.

Hardware performance counters used to monitor the LLC miss penalty are proposed and implemented in multi-core Godson-3 RTL and simulator. An algorithm is proposed and implemented that could recognize the application behaviors accurately and schedule them to suitable cores. [3]

Additional transistors and slower clocks means multi core designs and more parallelism required .For established processor design – increasing transistor density, speeding up clock rate, and lowering voltage have been blocked by a set of physical barriers: over heat produced, also much more power consumed and also much energy leaked, useful signal reduced by noise. Multi core designs are an accepted reaction to this condition. [4]

This architecture using many copies of the same core due to this improved total computational facility on single chip.Multi-core processors have enhanced performance and area characteristics than difficult single-core processors. [5] They propose and evaluate single-ISA heterogeneous multi-core architectures as a system to reduce processor power dissipation.

They present a technique for developing dense linear algebra algorithms that seamlessly scales to thousands of cores. It can be done with our plan called DPLASMA (Distributed PLASMA) that uses a new generic distributed Direct Acyclic Graph Engine (DAGuE).[9]

This architecture using many copies of the same core .Due to this improved total computational ability on a single chip . Multi-core processors have improved performance and area characteristics than complex single-core processors. [5] Assess single-ISA heterogeneous multi-core architectures as a method to decrease processor power dissipation.

IV. SHARING OF SHARED CACHE

The common nature of on chip caches is a property that is able to be exploited for performance gains. Data and instructions that be usually accessed by every cores in a shared method be able to exist rapidly reached by all cores.

This hardware performance feature leads to our first principle of promoting distribution in the shared cache. An operating system scheduler can select processes or threads that share data or instructions and co-schedule them to every run at the similar time within the same multi core processor so that they are able to make the most of the shared cache for sharing.

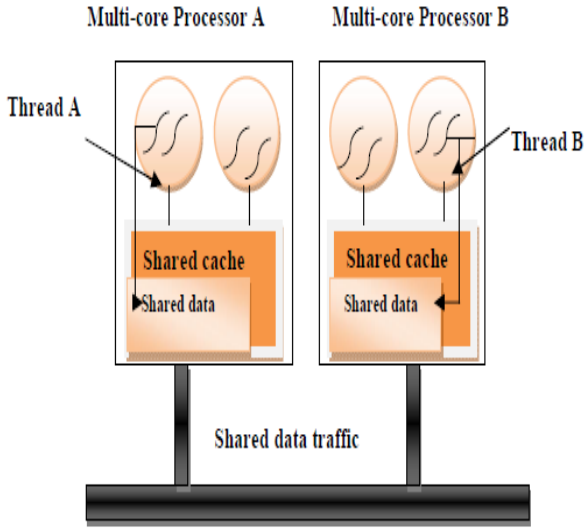

Fig. 6 Promoting sharing within a shared cache. Thread A and thread B are share data.

Thread B can be migrated to multi-core processor A so that the shared data is located within a single shared cache, resulting in more rapidly access by both threads, leading to improved performance.

A. Multiprocessor Parallelism

To enhance the number of instructions completed for every processor clock cycle, parallel instructions can be extracted from a sequential instruction stream as long as data dependencies between instructions.

If instructions B and C depend only ahead the result of instruction A, instruction D depends just upon the result of instruction B, and instruction E depends only upon the result of instruction C, then the most obtainable instruction-level parallelism for this instruction stream is 2.

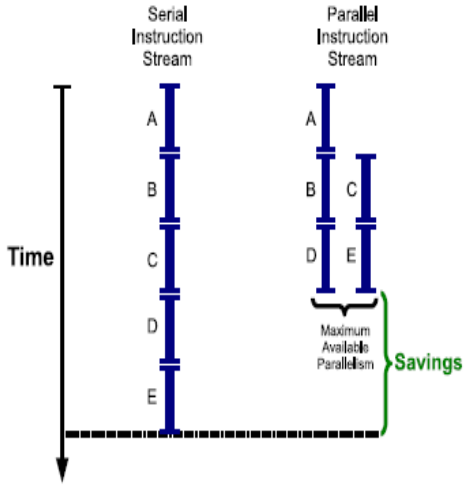

Fig. 7 Multiprocessor Parallelism

B. Moore's Law is Impotent

From an arithmetical point of view, performance inside a processor such as the Intel x86 family, can be calculated by the number of instructions completed per second (IPS: instructions per second), and is a result of the following.

$$\text{IPS}\left(\frac{\text{instruction}}{\text{second}}\right) = \text{IPS}\left(\frac{\text{instruction}}{\text{cycle}}\right) \quad \text{clock\_frequency}\left(\frac{\text{clock}}{\text{second}}\right)$$

Given N threads .To enhances the value of IPS, and thus performance, clock frequency can be increased, or IPC can be increased. Unfortunately, clock frequency now appears capped and IPC has imperfect potential due to difficulty in extracting instruction-level-parallelism (ILP) from a single serial instruction stream. Moore's Law no longer correlates to superior clock frequency values. Moore's Law does offer extra transistors that can help enhance IPC.

C. Thread-Level Parallelism

TLP describes the situation where there are many, independent threads of execution, which can be run at the same time inside a single processor. These multiple threads can come from either inside a single application or across multiple applications. This situation is like to the traditional multiprocessor parallelism where many threads are executed on many processors. Thread-level parallelism enables the IPC term in Equation to become the sum of the IPC of each thread, as exposed in Equation for N threads.-

$$\text{IPS} = \left(\sum_{n=1}^{N} \text{IPC n}\right) \times \text{clock\_frequency}$$

D. Memory Wall

Another matter limiting performance is the huge and growing disparity between processor speeds and main memory speeds in terms of both latency and bandwidth. This disparity is usually referred to as the memory wall. It will be reducing.

E. Exploiting Multiple Processors

There are many processors, caches, main memory banks, interconnects, and I/O devices .The operating system is responsible for the smart management of these common hardware resources. These hardware performance issues are usually addressed by the operating system by scheduling and memory management.

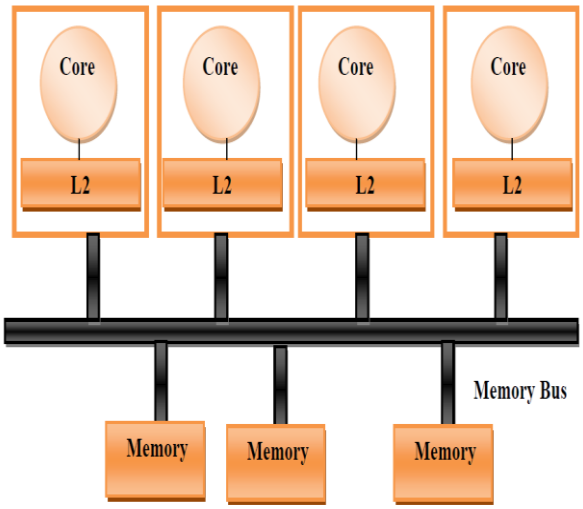

Fig. 8 SMP multiprocessor consists of several uni-core processors connected by a single shared memory bus.

F. Exploiting Simultaneous Multithreading

In simultaneous multithreading (SMT) processors, several micro-architectural hardware resources are shared among many threads of execution, leading to potential interference between threads. In adding to the processor pipeline resources, on-chip caches are too shared, which have the L1 instruction, L1 data, andL2 caches. The operating systems have to consider how to manage these shared hardware resources, by memory management and scheduling, in order to maximize application performance.

G. Exploiting Multiple Cores

On multi-core processors, the major hardware property that must be considered by the operating system is that there be able to be on-chip caches shared by many cores. This includes the L2 cache or, if it is presents the L3 cache. In difference, the L1 caches are private to each core, dissimilar in SMT processors. Another hardware property that should be considered by the operating system is that communication among cores is quicker than on traditional multi-chipped multiprocessors because every core is located on the identical chip, sharing the similar on-chip L2 cache

V. The Decoupled KILO-Instruction Processor (D-KIP)

In the D-KIP, two cores used to implement an application. The first core, the Cache Processor (CP), is small and fast, and executes all code depending only on cache accesses (high locality code). The CP runs forward as fast as possible, launching all loads with known addresses. Code to depend on memory accesses (i.e., low locality code) is processed through a secondary core, the Memory Processor (MP), which is proposed as a small in-order processor.

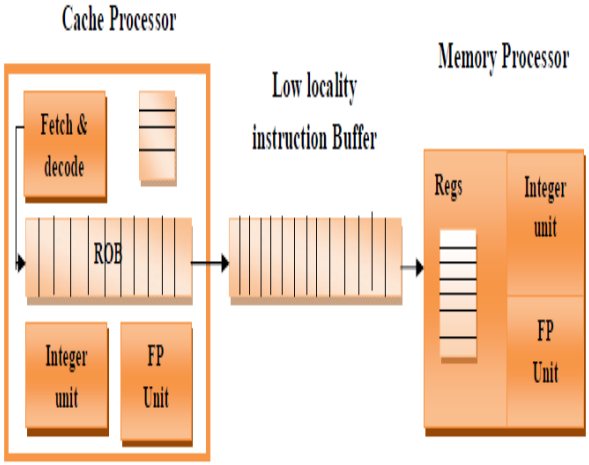

Fig. 9 Micro architecture of the D-KIP Processor

It executes low-locality code fetched from an in-order Low Locality Instruction Buffer (LLIB) that has earlier been filled in program order through the CP. Processor recovery is ensured by using checkpoints that are formed dynamically at the reorder buffer (ROB) output of the CP. Figure 9 shows a simplified overview of the D-KIP processor.

VI. Resizable Window with a Set of Memory Engines Memory Management

To partition the buffer into many in order smaller Buffers and provide each one with its own set of functional units. The buffers are then allocated round-robin to the cache

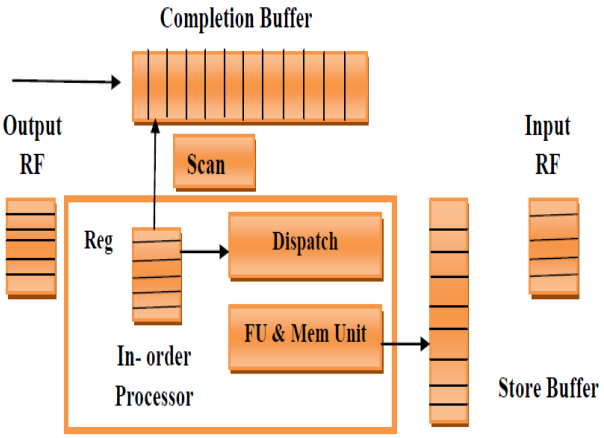

Fig.10 Architecture of a single Memory Engine

Processor as they are desired. In this scheme, instead of having a single memory engine we have a memory Processor consisting of a set of Memory Engines. Figure no.10 shows the Architecture of a single Memory Engine. Each of these memory engines is a copy of the multi-scan engine behavior a subset of the compressed instruction window.

Registers are passed between the engines with the input and output register files. After each scan the engine checks the output register file for recently generated registers .These registers are at that time sent over a network to the input register file of the next logical memory engine. Memory management is a critical part in high performance architectures.

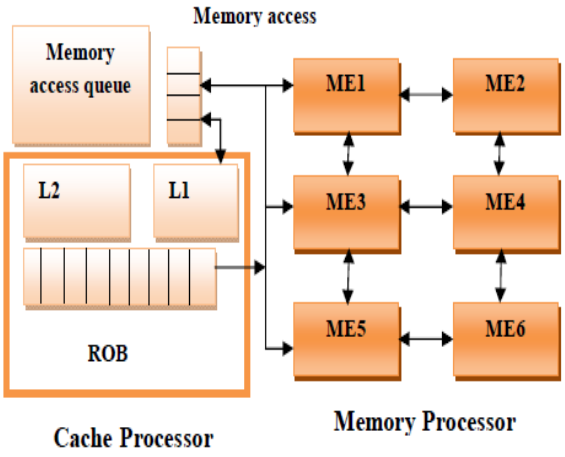

Fig.11 Processor with ME Network

VII. CONCLUSION

We presented a flexible multi-core (FMC) micro architecture able of with high performance. It high throughput used for identical parallel applications as well as high performance. Each core is extremely simple, thus our approach scales to a large number of cores, allowing for a capable and easy design. We consider this according to every method it gives the right path to provide best performance for workloads consisting of a large variety of applications. FMC performs this transparently to the programmer.

REFERENCES

[1] Miguel Pericas, Adrian Cristal, Francisco J. Cazorla, Ruben Gonzalez, Daniel A. Jimene and Mateo Valero "A Flexible heterogeneous Multi-Core Architectura".20 Parallel Architecture and Compiler Techniques 2007.

[2] Rakesh Kumar et. Al, "Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction", In Proceedings of the 36th International Symposium on Micro architecture, December 2003.

[3] Rangyu Deng et. "An Efficient Stream Memory Architecture for Heterogeneous Multi core Processor 2009.

[4] Shouqing Hao et. Al "Processes Scheduling on Heterogeneous Multi-core Architecture with Hardware Support" 2011.

[5] George Basilica et. Al "Flexible Development of Dense Linear Algebra Algorithms on Massively Parallel Architectures with DPLASMA"2011.

[6] H. Akkary, R. Raj war, and S. T. Srinivasan. Checkpoint processing and recovery: Towards scalable large instruction window processors. 2003.

[7] S. Balakrishnan, R. Raj war, M. Upton, and K. Lai. The impact of performance asymmetry in emerging multi core architectures. In Proc. of the Intl. Symp. On Computer Architecture, pages 506–517, June 2005.

[8] R. D. Barnes, S. Ryoo, and W. Mei W. Hwu. Flea-Flicker Multipass Pipelining: An Alternative to the High-Power Out-of-Order Offense. In Proc. of the 38th. Annual Intl. Symp. On Micro architecture, December 2005.

[9] Miguel Prices`, Adrian Cristal, Ruben Gonzalez, Daniela. Jimenez and Mateo Valero "A Decoupled KILO–Instruction Processor "in 2005.

[10] Miguel Prices et all" Chained In-Order/Out-of-Order Double Core Architecture" in 2006

[11] Francisco Javier Cazorla Almeida "Quality of Service for Simultaneous Multithreading Processors" in 2005.

[12] O. Mutlu, J. Stark, C. Wilkerson, and Y. N. Patt. Run ahead Execution: An alternative to very large instruction windows for out-of-order processors. In Proc .of the 9th Intl. Symp .on High Performance Computer Architecture, pages 129–140, 2003.

[13] M. Pericas, A. Cristal, R. Gonzalez, D. A. Jimenez, and M. Valero. A decoupled kilo-instruction processor. In Proc. of the 12th Intl. Symp. on High Performance Computer Architecture, February 2006.

[14] H. Zhou. Dual-core execution: Building a highly scalable single-thread instruction window. In Proc. of the 14th Intl. Conf .on Parallel Architectures and Compilation Techniques, September 2005.

[15] H. Akkary, R. Rajwar, and S. T. Srinivasan. Checkpoint processing and recovery: Towards scalable large instruction window processors. 2003.

[16] R.Kumar, V.Zyuban, and D.M.Tullsen. Interconnection in multi-core architectures: Understanding mechanisms, overheads, an scaling. In Proc of the 32ndIntl.Symp. On Computer Architecture, June 2005.

[17] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with CUDA. ACM Queue, 6(2):40{53, 2008.

[18] M. D. Lindeman, J. D. Collins, H. Wang, and T. H. Meng. Merge: a programming model for heterogeneous multi-core systems. In ASPLOS XIII, 2008.