International Journal of Enhanced Research in Science Technology & Engineering, ISSN: 2319-7463
Vol. 4 Issue 4, April-2015, pp: (331-339), Impact Factor: 1.252, Available online at: www.erpublications.com

Design of Multi-core System Using FPGA

Rashmi A. Jain[1], Dr. Dinesh Padole[2], Anjali Chavan[3], Rajaduty[4], Sumant[5]
[1,2,3,4,5]Electronics Engineering Department, D.Y.P.I.E.T. Pimpri Pune (M.S.) India

Abstract: Multi-core systems are dominating the processor market ranging from embedded to high-performance systems. Currently some processors integrate more than ten cores and, as the node shrinks in future technology generations, it is expected that the number of cores will continue increasing in future manufactured systems. To take advantage of the high number of cores efficient load balancing and scheduling policies or strategies are required. In addition, it remains a challenge to identify and productively program applications for these systems with a resulting substantial performance improvement. Multi-core system has wide utility in today's applications due to a reduced amount of power consumption and high performance. According to study of different architectures of multi-core system we have been presented two cores. It created by applying different partitioning strategy on the synchronous core architecture. On the other hand, the system designer must trade off performance versus power consumption, which is a major concern in current microprocessors. Therefore, current design must focus on new architectures or architectural mechanisms addressing this tradeoff.

Keywords: Synchronous core, general purpose processor core, low power, Mutli-core, SoC.

I. INTRODUCTION

A multi-core processor is a single computing component with two or more independent actual central processing units (called "cores"), which are the units that read and execute program instructions. [1] The instructions are ordinary CPU instructions such as add, move data, and branch, but the multiple cores can run multiple instructions at the same time, increasing overall speed for programs amenable to parallel computing. Manufacturers typically integrate the cores onto a single integrated circuit die (known as a chip multiprocessor or CMP), or onto multiple dies in a single chip package. Multi-core processors are widely used across many application domains including general-purpose, embedded, network, digital signal processing (DSP), and graphics. The improvement in performance gained by the use of a multi-core processor depends very much on the software algorithms used and their implementation. In particular, possible gains are limited by the fraction of the software that can be run in parallel simultaneously on multiple cores; this effect is described by Amdahl's law. In the best case, so-called embarrassingly parallel problems may realize speedup factors near the number of cores, or even more if the problem is split up enough to fit within each core's cache(s), avoiding use of much slower main system memory. Most applications, however, are not accelerated so much unless programmers invest a prohibitive amount of effort in re-factoring the whole problem. [3]

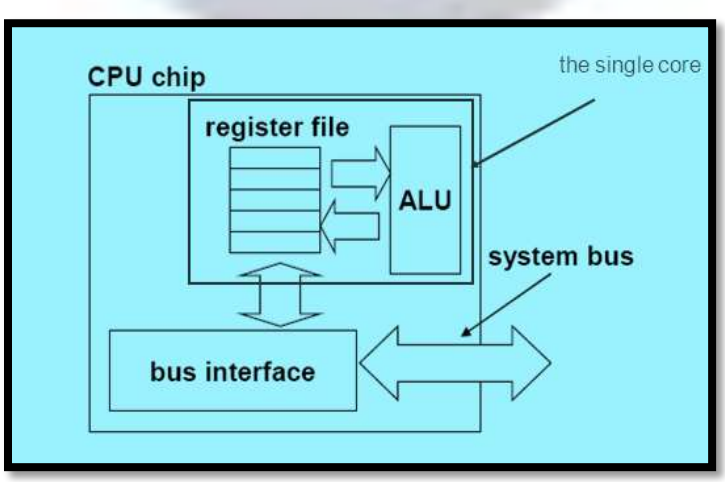


Fig 1 Single-core CPU chip

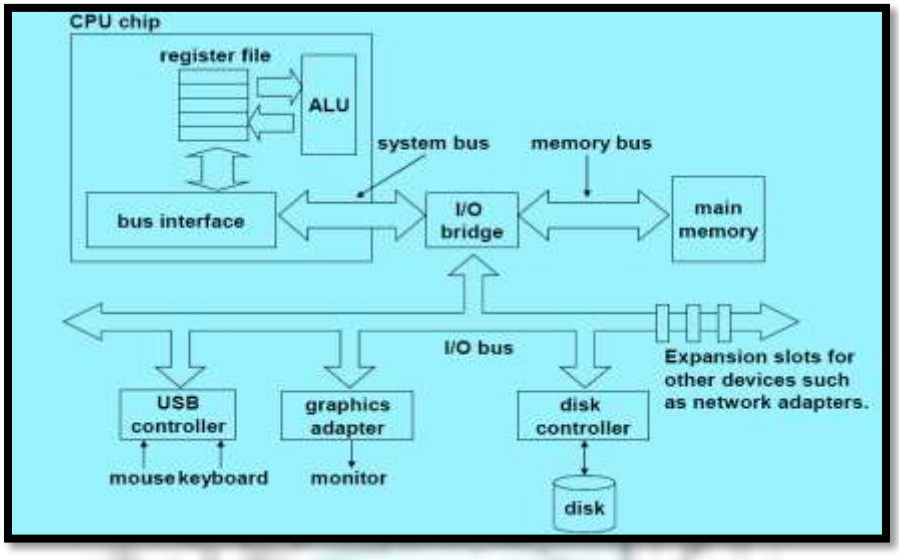


Fig 2 Multi-core processor

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.

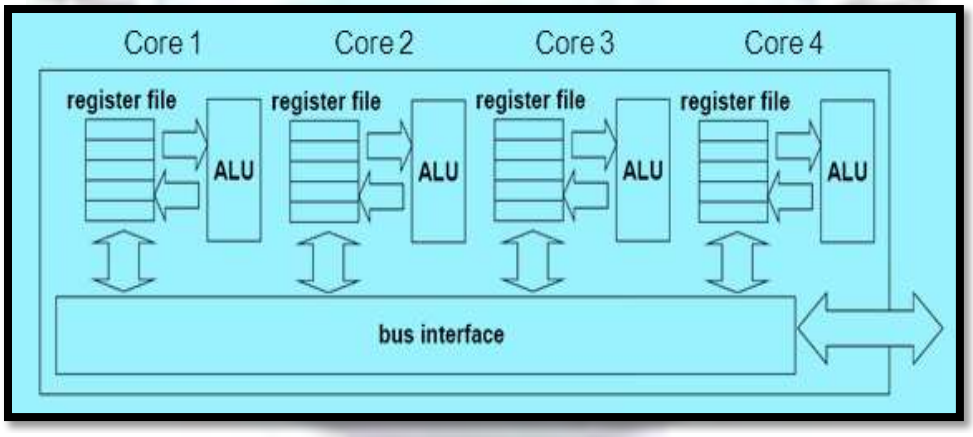


Fig 3 Multi-core CPU chip

II. RELATED WORK

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. 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.

III. THEORY

A. Synchronous core Design:

Till date microprocessors or core available in market are all built using synchronous technology? In synchronous systems there is a centralized clock-pulse originator. Each part of the core has to be connected to the same clock-signal. I/O Pin Description of synchronous core given in table no.1.The potential advantages of asynchronous design, one might question why synchronous systems predominate. The idea of designing this core is taken from [18] with several modifications in the coding as well as in the architecture to get the required, errorless simulation and synthesis results. Fig.1 shows the architectural block diagram of the 8-bit synchronous core implemented in VHDL.

B. Types of multi-core system

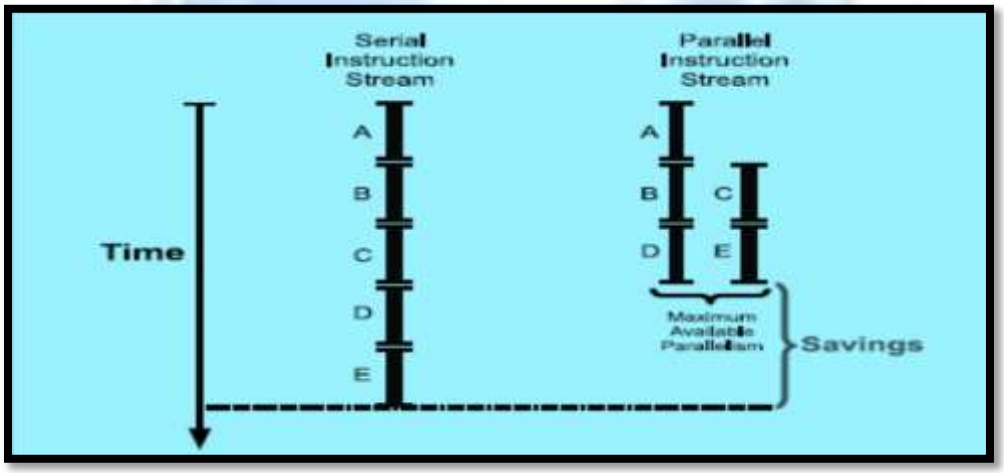
1. Homogeneous multi-core systems-Homogeneous multi-core systems have only identical cores. Some system use one core design repeated consistently known as homogeneous multi-core systems.
2. Heterogeneous multi-core systems-Heterogeneous multi-core systems have cores which are not identical. It means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

C. 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.

D. 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.

E .Terminology

The terms multi-core and dual-core most commonly refer to some sort of central processing unit (CPU), but are sometimes also applied to digital signal processors (DSP) and system-on-a-chip (SoC). The terms are generally used only to refer to multi-core microprocessors that are manufactured on the same integrated circuit die; separate microprocessor dies in the

same package are generally referred to by another name, such as multi-chip module. This article uses the terms "multi-core" and "dual-core" for CPUs manufactured on the same integrated circuit, unless otherwise noted.

In contrast to multi-core systems, the term multi-CPU refers to multiple physically separate processing-units (which often contain special circuitry to facilitate communication between each other). The terms many-core and massively multi-core are sometimes used to describe multi-core architectures with an especially high number of cores (tens or hundreds).[4] Some systems use many soft microprocessor cores placed on a single FPGA. Each "core" can be considered a "semiconductor intellectual property core" as well as a CPU core.

Table1. I/O Pin Description of synchronous core

Signals	Width(bits)	I/O	Name/Function
Input_mpu	8	I	Input port of the core module
Clk_ mpu	1	I	Clock for the core module
Rst_ mpu	1	I	Active high reset for the core module
Output_mpu	8	O	Output for the core module

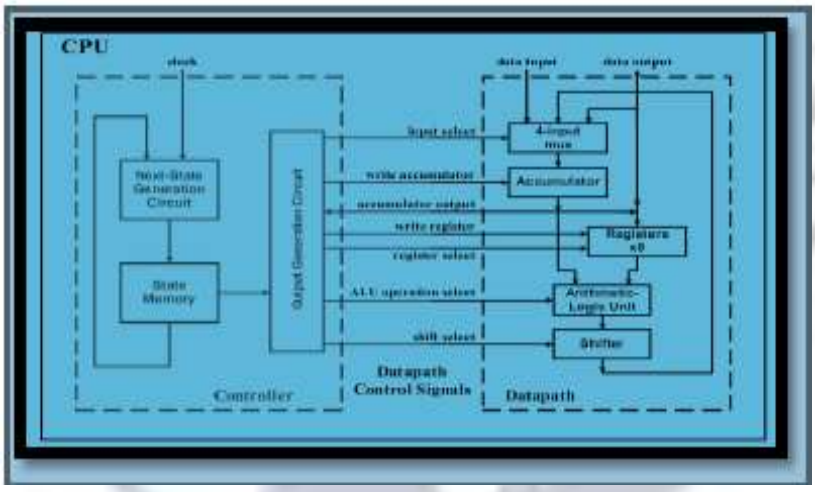


Fig 4 8-bit Synchronous core architecture

IV. ADVANTAGES

- Difficult to make single-core clock frequencies even higher
- Deeply pipelined circuits:
  - heat problems
  - speed of light problems
  - difficult design and verification
  - large design teams necessary
  - server farms need expensive air-conditioning
- Many new applications are multithreaded
- General trend in computer architecture (shift towards more parallelism)
- Having a multi-core processor in a computer means that it will work faster for certain programs.
- The computer may not get as hot when it is turned on.
- The computer needs less power because it can turn off some sections if they aren´t needed.
- More features can be added to the computer.
- The signals between different CPUs travel shorter distances, therefore they degrade less

V. DISADVANTAGES

- They do not work at twice the speed as a normal processor. They get only 60-80% more speed.
- The speed that the computer works at depends on what the user is doing with it.
- They cost more than single core processors.
- They are more difficult to manage thermally than lower-density single-core processors.
- Not all operating systems support more than one core.
- Operating systems compiled for a multi-core processor will run slightly slower on a single-core processor.

VI. APPLICATION

- Database servers
- Web servers (Web commerce)
- Compilers
- Multimedia applications
- Scientific applications, CAD/CAM
- In general, applications with
  Thread-level parallelism
  (as opposed to instruction-level parallelism)
- Editing a photo while recording a TV show through a digital video recorder
- Downloading software while running an anti-virus program
- "Anything that can be threaded today will map efficiently to multi-core"
- BUT: some applications difficult to parallelize

VI. PROPOSED 8-BIT SYNCHRONOUS MPU

The design of a microprocessor can be divided into two main parts: control unit and datapath. In designing a CPU, we must first define its instruction set and how the instructions are encoded and executed. Once we have decided on the instruction set, we can proceed to designing a datapath that can execute all the instructions in the instruction set. Finally, we can design the control unit. The control unit acts as a finite- state machine which cycles through three main states: 1) fetch an instruction; 2) decode the instruction; and 3) execute the instruction. The control unit performs these steps by sending the appropriate control signals to the datapath blocks.

Instructions in a program are usually stored in external memory, so in addition to the CPU, there is external memory that is connected to the CPU via an address bus and a data bus. Hence, step 1 (fetch an instruction) usually involves the control unit setting up a memory address on the address bus and telling the external memory to output the instruction from that memory location onto the data bus. The control unit then reads the instruction from the data bus. To keep the design simple, instead of having external memory, the memory is put directly inside the CPU and implemented simply as a 16-byte array. For step 2 (decode the instruction) the control unit extracts the opcode bits from the instruction and determines what the current instruction is by jumping to the state that has been assigned for executing that instruction. Once in that particular state, the finite-state machine performs step 3 by simply asserting the appropriate control signals for controlling the datapath to execute that instruction. Table 1 shows the instruction set that the proposed microprocessor can execute.

Table 2. Instruction set

INSTRUCTION	ENCODING	COMMENT
LDA A, rrr	0000 0rrr	Load accumulator from register
AND A, rrr	1000 0rrr	Acc. AND register
OR A, rrr	1001 rrr	Acc. OR register
ADD A, rrr	0001 0rrr	Acc. + register
SUB A, rrr	0010 0rrr	Acc. – register
NOT A	1010 0000	Invert acc.
INC A	0011 0000	Increment acc.
DEC A	0100 0000	Decrement acc.
SHFR A	1011 0000	Shift acc. right
ROTR A	1100 0000	Rotate acc. Right
HALT	1111 0000	Halt execution

To design the processor, first the behavior of the different datapath blocks (i.e. 4:1 mux, 8-bit accumulator,8*8 register file, 8-bit ALU, 8-bit shifter) was described and then using structural modeling the different datapath components were interconnected. After that control unit was coded as finite–state machine in behavioral modeling. Finally the top level consists of structural modeling description to interconnect control and datapath units. The behavioral simulation results were verified for all the instructions.

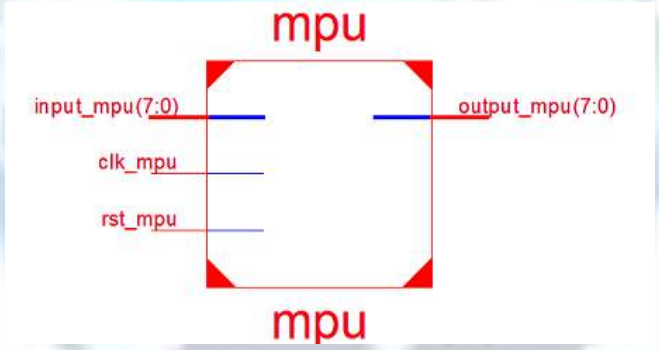


Fig 5. Top level entity of the 8-bit processor unit

VII. MULTI-CORE TECHNOLOGY

In consumer technologies, multi-core is usually the term used to describe two or more CPUs working together on the same chip. Also called multi-core technology, it is a type of architecture where a single physical processor contains the core logic of two or more processors. These processors are packaged into a single integrated circuit (IC). These single integrated circuits are called a die. Multi-core can also refer to multiple dies packaged together. Multi-core enables the system to perform more tasks with a greater overall system performance. Multi-core technology can be used in desktops, mobile PCs, servers and workstations. Contrast with dual-core, a single chip containing two separate processors (execution cores) in the same IC.

VIII. RESULTS OF SIMULATION AND SYNTHESIS

The VHDL code for synchronous processor and Multi-core system are synthesized using Xilinx Synthesis Tool (XST) and the functionality is verified using ISE simulator (ISim). The VHDL code for synchronous and GALS is synthesized using Xilinx Synthesis Tool (XST) and the functionality is verified using ISE simulator (ISim).

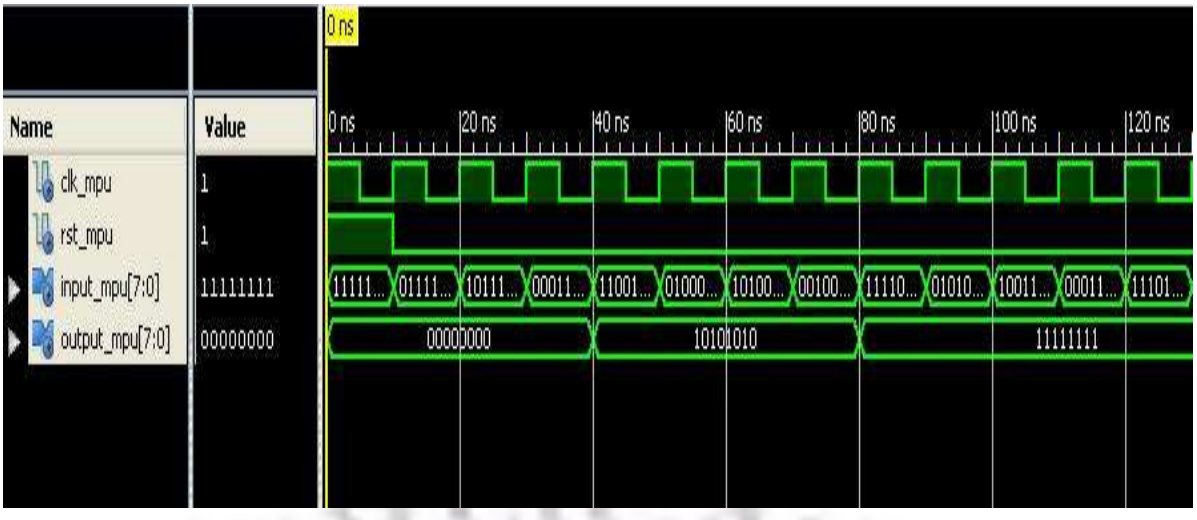


Fig 6. ADD instruction of core

```
SYNCHRONOUS DESIGN
Timing Summary:
---------------
Speed Grade: -5

   Minimum period: 6.959ns (Maximum Frequency: 143.698MHz)
   Minimum input arrival time before clock: 4.304ns
   Maximum output required time after clock: 6.318ns
   Maximum combinational path delay: No path found
```

Fig 7. Timing summary of Synchronous processor

```
SYNCHRONOUS DESIGN
Device utilization summary:
---------------------------

Selected Device : 3s400fg320-5

 Number of Slices:                       31  out of   3584     0%
 Number of Slice Flip Flops:             27  out of   7168     0%
 Number of 4 input LUTs:                 59  out of   7168     0%
    Number used as logic:                51
    Number used as RAMs:                  8
 Number of IOs:                          18
 Number of bonded IOBs:                  10  out of    221     4%
 Number of GCLKs:                         1  out of      8    12%
```

Fig 8. Device utilization summary of Synchronous processor

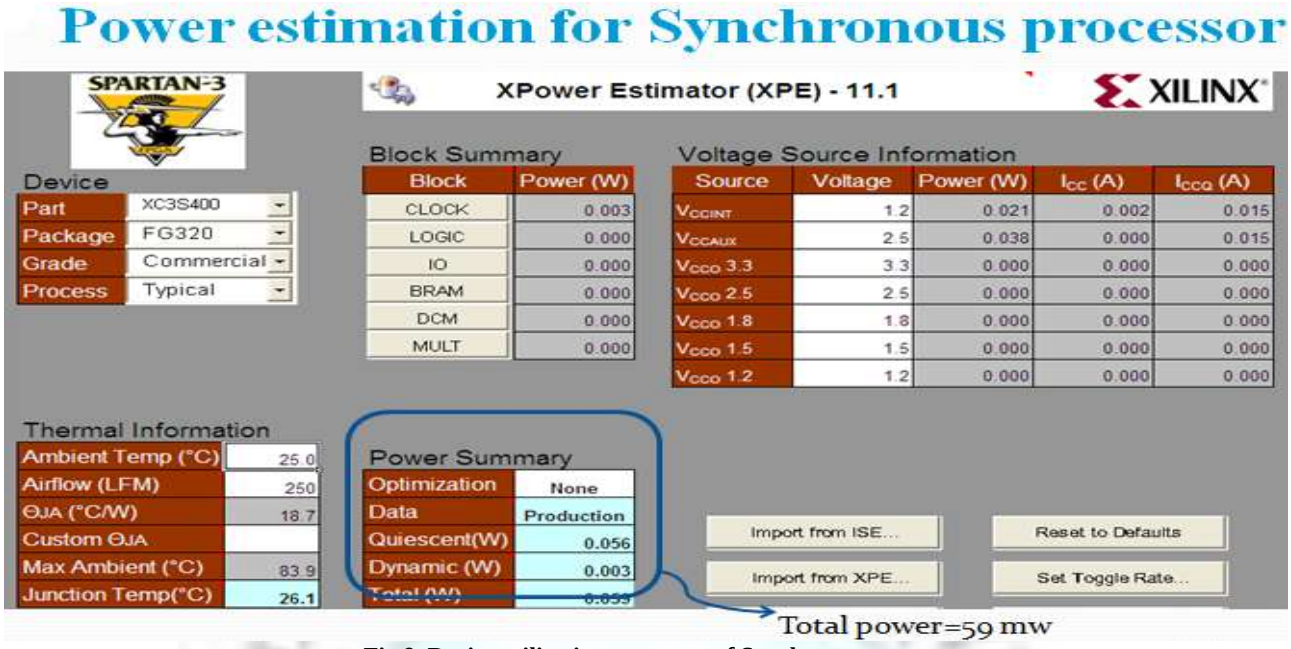


Fig 9. Device utilization summary of Synchronous core

CONCLUSION

Multi-core chips an important new trend in computer architecture. Several new multi-core chips in design phases. Parallel programming techniques likely to gain importance. We presented a multi-core 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. Multi-Core 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]. Rashmi Jain et al "Globally Asynchronous Locally Synchronous Design Based Heterogeneous Multi-core System"(eds.), ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of CSI - Volume I, Advances in Intelligent Systems and Computing 248, DOI: 10.1007/978-3-319-03107-1_81, © SpringerInternationalPublishingSwitzerland2014.
[18]. 1.Rashmi A Jain 2.Dr. Dinesh Padole "Scalable and Flexible heterogeneous multi-core system" (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 3, No. 12, 2013 .