International Journal of Electrical and Computer Engineering (IJECE)
Vol. 3, No. 3, June 2013, pp. 423~428
ISSN: 2088-8708

Design of High Performance and Low Power Simultaneous Multi-threaded Processor

Krishan Arora*, Paramveer Singh Gill*, Parul Mehra**
* Assistant Professor Departement of Electronics and Electrical Engineering, Lovely ProfessionalUniversity, Punjab
**Phd Scholar, Departement of Commerce, CMJ University, Meghalaya

Article Info

Article history:

Received Mar 31, 2013
Revised May 14, 2013
Accepted May 27, 2013

Keyword:

Muti-threaded Prosessor
Hardware thraeding
SMT
Instruction scheduling
Single Threaded Processor

ABSTRACT

In this paper, we present the design of a High Performance Multi-Threaded Processor. Processing of high quality images is inevitable in applications such as, HD TV, Gaming Multimedia, etc. which require a great processing power with low power consumption. This can be achived with multi-threaded processors which optimally utilises the Functional Units (Fus). The speed of processing is as good as multi-core processors with lesser area. A conflict resolver (CR) is designed for scheduling the instructions, which involves allocation of Fu. The data move instructions are in majority in any of the programs; the corresponding logic blocks are replicated and speed of execution is further improved. We illustrated for two-threaded processorHowever, it is possible to extend the design for any number of threads by suitably redesigning the CR, and also replicate Transfer Logic and CPU Registers.

Corresponding Author:

Krishan Arora
Assistant Professor
Departement of Electronics and Electrical Engineering,
Lovely Professional University,Punjab
Email: er.krishanarora@gmail.com

1. INTRODUCTION

The extreme developments in entertainment, gaming, medical imaging and HDTV along with electronics systems. Electronic systems design is getting more and more complex. The system requires very high speed processing, at the same time power consumption is also stretching its limitations. To meet this contradicting requirements the solution is in the efficient and effective embedded processor design. One of the promising methods is “Simultaneous multithreading” (SMT), which takes super-threading to the next level in the high performance processor design. It is super-threading without the restriction, i.e. all the instructions issued by the front end on each clock are from the same thread. In SMT, instructions are executing on different threads simultaneously. So we get better utilization of functional units. The issue of scheduling is properly managed by a special hardware unit called Conflict analyzer, which takes part of the opcode, i.e. 3 most significant bits.

Although SMT might seem like a pretty large departure from the kind of conventional, process-switching multithreading done on a single-threaded CPU, it actually doesn't add too much complexity to the hardware. This is done by dividing up the processor's architectural resources into two types: Replicated and Shared.

2. RESEARCH METHOD

The potential for achieving a significant increase in throughput on a superscalar by using simultaneous multithreading (SMT) was first demonstrated in 1995 by Tullsen of University of washington

[1]. SMT is a technique that allows multiple independent threads or programs to issue multiple instructions to a superscalar's functional units. Then the Dynamic Multithreading Processor [3] presents a simultaneous multithreading pipeline to increase processor utilization, except that the threads are created dynamically from the same program. In [4] packet processor utilizes multiple multithreaded processing engines to support this parallelism in a design that supports 256 simultaneous threads in eight processing engines. Each thread has its own independent register file and executes instructions formatted to a general purpose ISA, while sharing execution resources and memory ports with other threads. The processor is optimized to sacrifice single threaded performance, so that a design is achieved that is realizable in terms of silicon area and clock frequency. In [5] reexamine the issue of nondeterministic processors, simplifying previous designs using a multithreaded architecture. In [6] presents a cache-based architecture support for Speculative Simultaneous multithreading which can efficiently handle larger threads. Speculative multithreading is a technique that has been used to improve single thread performance. In [7] presents the Mitosis framework, which is a combined hardware-software approach to speculative multithreading, even in the presence of frequent dependences among threads. Speculative multithreading increases single-threaded application performance by exploiting thread-level parallelism speculatively, i.e., executing code in parallel, even when the compiler or runtime system cannot guarantee that the parallelism exists. In [8] describes a transaction-level simulation model of a multi-core, multithreaded architecture and the usage of an application model that generates synthetic single or multi-threaded execution traces to drive the simulation. In [9] explains a conceptual architecture developed for modeling multi-threaded processors and the new functionality Model to support multithreaded processors.

Our work is different from above in two ways: (1) A new Single Treaded Processor is introduced as base platform for implementing Multi-Threading, It fetch the op-code and data simultaneously and decodes and executes that in single machine cycle i.e. it takes only two clock periods. (2) By virtue of this we got simple hardware, which consume very less power and Silicon area. It is because we are fetching op-code and data from different pins so we do not need any internal circuit which is require to redefining pins from machine cycle to machine cycle. That circuit is composed of complex FSMs, so by eliminating that our switching power also gets reduced and execution speed increased by many folds. Also it is proved form the initial studies that the methodology can be extended to more than two threads.

3. SINGLE THREADED PROCESSOR ARCHITECTURE

We first implemented the Single Threaded Processor which has following features:

1. It can execute 45 Arithmetic, Logical, Data transfer and Looping Instructions.
2. All the instructions are taking only 2 clock periods to execute.
3. It can communicate with 2K IO devices.
4. It has 256 bytes internal Cache memory.
5. It can point to 2K external memory.
6. The maximum Clock Frequency is 264MHz.

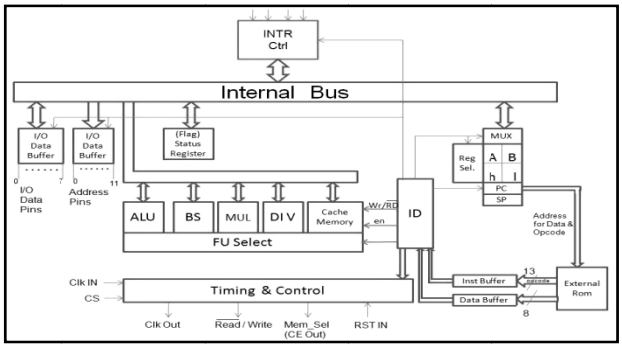

Figure 1. Architectural Diagram of Single Threaded Processor

4. IMPLEMENTATION OF MULTI-THREADED PROCESSOR

Although SMT might seem like a pretty large departure from the kind of conventional, process-switching multithreading done on a single-threaded CPU, it actually doesn't add too much complexity to the hardware. This is done by dividing up the processor's architectural resources into two types: Replicated, and Shared. Let's take a look at which resources fall into which categories for my Multi-threaded processor:

Replicated	Shared
▪ All General Purpose Registers ▪ PC and IR ▪ Return Stack predictor ▪ Flag Register ▪ Transfer logic to implement the data transfer instructions	▪ Cache Memory ▪ Execution Units ▪ Instruction Decoder ▪ Functional Units ▪ IO pins

We implemented Two Thread processor which can run two Threads of a program on one copy of Fus.(i.e. in our processor there is only one ALU, one Multiplier, one Divide, one Barrel shifter). The Instruction Decoder and Conflict Resolver efficiently utilizes the Fus. The arhitecture of Multi-Threaded Processor is shown in Figure 2.

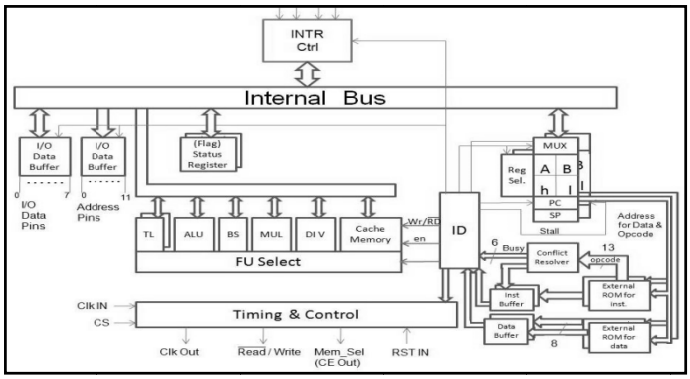

Figure 2. Architecture of Multi-Threaded Processor

4.1. Conflict Resolver

Conflict Resolver is nothing but a simple decoder which identify by the MSBs of op-codes of next Instruction that which functional unit will be used by threads in the next Instruction. So it will reserve the Functional Unit used in the next Instruction by Thread-1 for Thread-1and allow the Thread-2 to execute its instruction on the rest of Functional Units. If in the case both threads requires same functional unit in the next instruction then conflict resolver issue a signal Stall=1, which will stop the program counter of thread-2 to increment and do not allow its instruction to execute while do nothing with Thread-1(i.e. Instruction of Thread-1 will execute normally). The same cycle is repeat again and again so by this conflict resolver is utilizing functional units efficiently without any error in the output. All the above is shown clearly by the flow chart in the Figure 3.

Table 1. Representation of conflicting cases and the respective FUs

Case No	OP_{12}	OP_{11}	OP_{10}	Conflicting FU
1	0	0	1	ALU
2	0	1	0	Divider
3	0	1	1	Barrel Shifter
4	1	0	0	Multiplier
5	1	1	0	Cache
6	1	1	1	IOB

We replicated the Transfer Logic (TL) (which takes care of all data transfer between Register-2, Register-Memory, Register-I/Os etc) .It means each thread has its own copy of TL. It is because in a program most of instructions are related to data transfer, so by providing each thread its own copy of TL it will not stall other thread for TL related Instructions, so the execution will be fast and smooth.

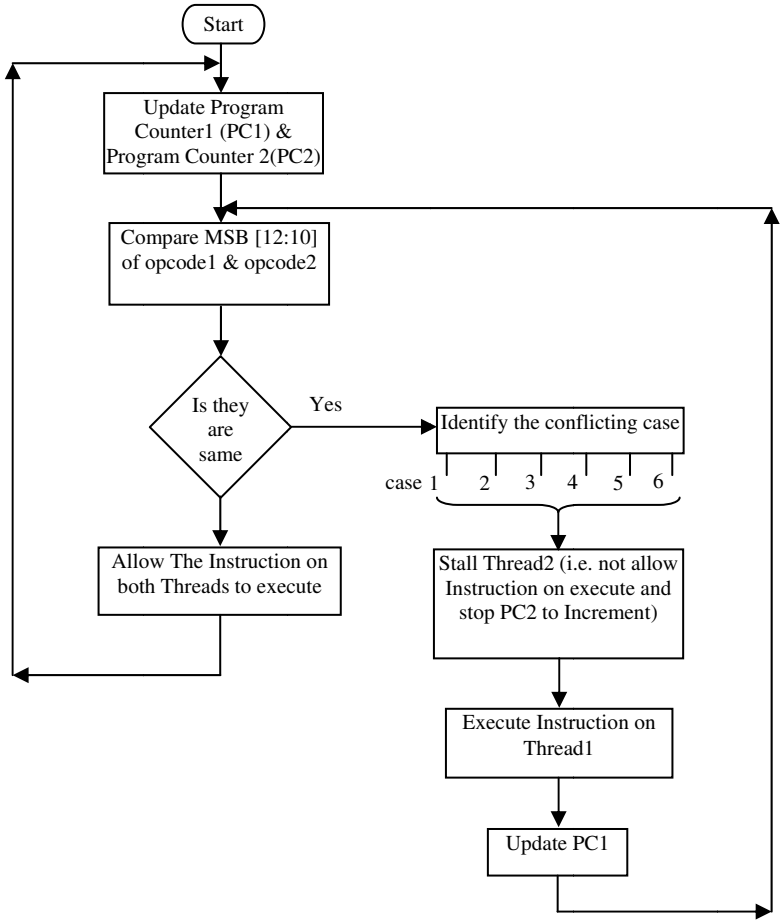

Figure 3. Flow Chart - Conflict Revolver

5. RESULTS AND ANALYSIS

A total of three Architecture (Viz. Single Threaded, Multi-Core, in which all the resources were considered to be replicated and Multi-Threaded, in which partially replicated and others were shared) of Processors are compared on Area, Power and Throughput bases and results are shown below:

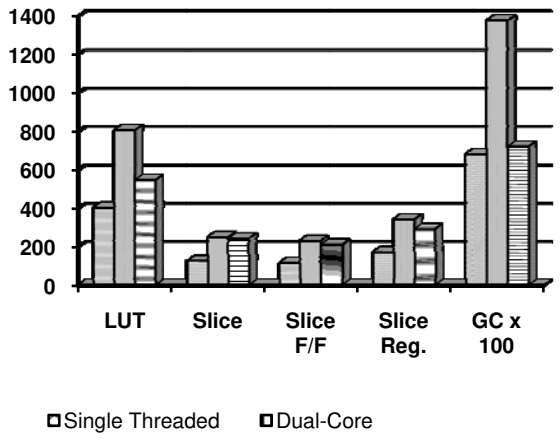

Figure 4. Area comparison in terms LUT, Slice, F/F, Registers and Gate count

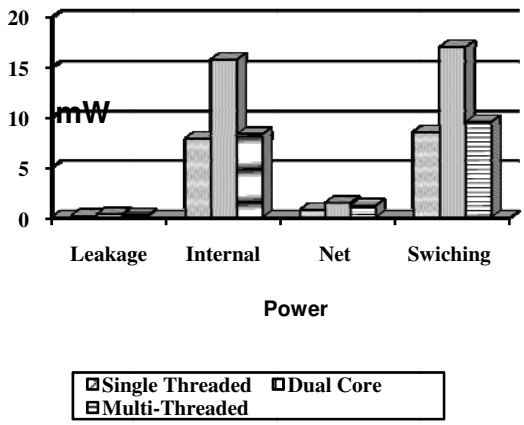

Figure 5. Power consumption in mW

So as we can see from the above graphs that the gate count and power consumption of Dual core processor is just double as that of Single Threaded Processor. But there is lot of area and power saving in case of Multithreaded Processor as compare to Dual Core Processor. Its is mainly because in case of Multi-Threaded the all functional units are not replicated but they are shared, while in case of Dual-Core processor whole the architecture get replicated.

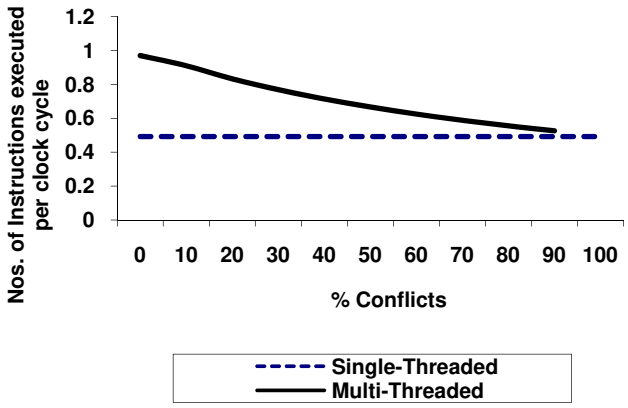

Figure 6. Performance with respect to dependencies among the threads

Here we compared performance of Single-Threaded Processor and Multi-Threaded Processor on the bases of Instructions executed per clock cycle for 50 Instructions program per Thread (i.e. total of 100 instructions). For 0 and 2% conflicts our Multi-Threaded Processor performs 197% compare to Single-Threaded Processor (i.e. nearly like two Single-Threaded Processors). As percentage of conflicts increases the performance of Multi-Threaded processor starts decreasing nearly linearly. Since for 50 instructions per thread the maximum possible conflicts can be 94%. At 94% conflict Multi-Threaded will perform 104.63%.But as we know most of the Instructions in a program is Data-transfer and Jump, so in most of cases the conflict will be will be varies from 30-60% and correspondingly our Multi-Threaded processor performance varies from 156.15-126.88% compare to Single-Threaded Processor.

6. EXPERIMENTATION

The processor is modelled using Verilog HDL and its functionality is verified by using ModelSim-XE III 6.2g. It is then synthesized with Xilinx ISE-9.1i. Finally the design is mapped on Vertex5-XC5VLX110T platform.

7. FUTURE SCOPE

The presented Multi-Threaded processor architecture can be extended to any number of threads by suitably redesign the CR, also replicate transfer logic and CPU Registers as many as threads.The usual limitation on the number of threades is number of functional units used in the design. Incase the number of threads exceeds the number of functional units, the threades has to wait more to get perticular functional unit to execuete its instruction, so the perforemence of Multi-Threaded processor will degrade greatly . But there is also scope we can either go for combination of Multi-Core and Multi-threaded processor, in which there will be multi-cores of processor on single chip and each core will have perticular number of threads.

REFERENCES

[1] Dean M. Tullsen, Susan J Eggers, Henry M Levy. *Simultaneous Multithreading: Maximizing On-Chip Parallelism.* ICSA'95, Santa Margherita Ligure Italy. 1995: 392–403.
[2] Susan J Eggers, Joel S Emer, Henry M Levy, Jack L Lo, Rebecca L Stamm, Dean M Tullsen. *Simultaneous Multithreading: A Platform for Next-Generation Processors.* Journal of IEEE Micro. 1997: 12-17.
[3] Haitham Akkary, Michael A Driscoll. *A Dynamic Multithreading Processor.* 31st Annual ACM-IEEE International Symposium on Microarchitecture. 1998: 226-236.

[4] O'Melveny, Myers LLP, Kayamba, Inc. *A Massively Multithreaded Packet Processor.* Presented at NP2: Workshop on Network Processors, held in conjunction with The 9th International Symposium on High-Performance Computer Architecture, Anaheim, California. 2003: 1-11.
[5] P Leadbitter, D Page, NP Smart. Nondeterministic Multithreading. *IEEE Transactions on Computers*. 2007; 56(7): 992-998.
[6] Venkatesan Packirisamy, Shengyue Wang, Antonia Zhai, Wei-Chung Hsu and Pen-Chung Yew. *Supporting Speculative Multithreading on Simultaneous Multithreaded Processor.* Y Roberts et al. (Eds.): HiPC. 2006: 148-158.
[7] Carlos Madriles, Carlos Garcı´a-Quin˜ones, Jesu´s Sa´nchez, Pedro Marcuello, Dean M. Tullsen. *Mitosis: A Speculative Multithreaded Processor Based on Precomputation Slices.* IEEE Transactions on Parallel and Distributed Syatems. 2008; 19(7): 914-925.
[8] Nicholas Ma, Naraig Manjikian, Subramania Sudharsanan. *Modeling and Simulation of Multicore multithreaded Processor Architecture in System C.* CCECE. 2008: 1155-1160.
[9] David Burgart. *Modeling Multi-Threaded Processors.* White paper TeamQuest. 2008: 1-10.

BIOGRAPHIES OF AUTHORS

Krishan Arora is presently working as Assistant Professor in the Department of Electronics and Electrical Engineering in Lovely Professional University, Phagwara (Punjab). He has completed his B.Tech (Electrical & Electronics Engg.) from Punjab Technical University,Punjab and M.Tech (Electrical Engg.) from Punjab Technical University, Punjab. His area of research is Power System, Electrical Machines and Power Electronics.

Paramveer Singh Gill is presently working as Assistant Professor in the Department of Electronics and Electrical Engineering in Lovely Professional University, Phagwara (Punjab). He has completed his B.Tech (Electronics & Communication Engg.) from Punjab Technical University,Punjab and M.Tech (VLSI Design) from VIT Vellore. His area of research is Processor Design.

Parul Mehra is Phd Scholar from Department of Commerce in CMJ University, Meghalaya. She has completed her Master of Commerce from Hindu College, Amritsar and Bachelors of Education with specialization in Commerce and Economics from Khalsa college of Education Amritsar. She has cleared UGC-NET with specialisation in Commerce in DEC.2011 in First Attempt.