Nurture IDR Segmentation and Multiple Instruction Queues in Superscalar Pipelining Processor

J.Nandini meeraa[1], N.Indhuja[2], S.Devi Abirami[3] and K.Rathinakumar[4]

**[1] Computer Science and Engineering, Narasu's Sarathy Institute of Technology
Salem, Tamil Nadu, India**

**[2] Computer Science and Engineering, Narasu's Sarathy Institute of Technology
Salem, Tamil Nadu, India**

**[3] Computer Science and Engineering, Narasu's Sarathy Institute of Technology
Salem, Tamil Nadu, India**

**[4] Electronics and Communication Engineering, Narasu's Sarathy Institute of Technology
Salem, Tamil Nadu, India**

Abstract

This paper proposes a model which improves the speed of the pipelining mechanism therefore increasing the speed of the processor. Superscalar operation is used to get maximum throughput from the processor using the pipelining concept. This proposal can be considered as the advancement of the super scalar property in pipelining which presently exists. We introduce a concept, using multiple instruction queues and a new unit called as Identifier unit. The Identifier unit is designed as having the ability of the identifying the type of the instruction which is being fetched and separating it based on its types thus creating separate segments of execution, which in turn increases the speed of the processor. Moreover we have implemented separate decoding and the executing unit for each type of the instruction segments.

Keywords: *Pipelining, Superscalar property, Identifier unit and Multiple Instruction queue.*

1. Introduction

In a computer system the work of the processor is to execute the instructions given by the user. The processor executes the instructions in different fashions. Olden days the machine level instructions were executed in serial fashion which took lot of time the Pipelining mechanism is used achieve high execution rate in a processor. Superscalar operation is used to improve the speed of the processor.

Nurture IDR segmentation and multiple instruction queues in superscalar pipelining processor, is a model which is the modification of superscalar property. This model can enhance the speed of the pipelining processor. The core idea of this process is segmentation of the machine instructions which is being fetched from the fetch unit based on its type and storing it into separate instruction queue for each type, thus forming multiple instruction queues.

Pipelining is the first concept which is to be known in detail to understand the working of the Nurture IDR segmentation and multiple instruction queues in superscalar pipelining processor. Moreover the superscalar operation and the processing of the instruction through it, is important because it is the base of the proposed system. The implementation of Nurture IDR segmentation and multiple instruction queues in superscalar pipelining processor is aimed at increasing the speed of the superscalar pipelined processor. The working of each unit of the proposed system such as fetch unit, identifier unit, decode unit, execute unit and write unit are explained with sufficient information of its design and working. The new concept of Nurture IDR segmentation and multiple instruction queues in superscalar pipelining processor will definitely create an evolution in the speed of the superscalar pipelined processor.

2. Pipelining

The olden processors used the sequential technique to execute the instructions. The instructions were fetched and executed one by one in serial fashion, which took lot of time. Once the instruction is being fetched by the fetch unit it is executed by the execution unit during the execution of the instruction the fetch unit is ideal.

The following diagram Fig: 1 shows the execution of the instructions in the sequential execution of instruction

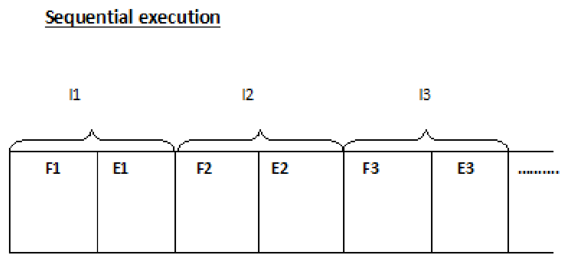

Fig: 1 sequential technique of execution

This order of execution had many disadvantages which lead to the invention of the Pipelining mechanism. Pipelining is the concept which has separate stages called as the fetch unit, decode unit and execute unit and the write unit [1].By using the pipelining the time taken to execute an instruction is faster the sequential order of execution. This made the pipelining mechanism to a successful fashion of execution of the machine instruction in the processor. This mechanism is being used in present processors.

Pipelined instruction processing is a basic technique used in the design of micro-architectures [2]. Pipelining is method in which the instructions are executed in separate stages such as the fetch unit, decode unit and execute unit and the write unit. The fetch unit will actually read the instruction from the memory unit hence fetching it and after which the instruction reaches into the decoding unit where the instruction is decoded from high level language to low level language [1]. Then it enters into the execution unit where the ALU performs the arithmetic and the logical operations. Finally the executed instruction in sent to the write unit to write the results. Consider the following example to understand the working mechanism of pipelining concept. In a PC manufacturing company, there are different units of people working to manufacture a single PC. First the design unit plans the design of the PC.

Then supplier unit supplies the hardware material needed, then the assembling unit do the assembling work of the hardware which is followed by the testing unit so on. Once the designing is over by the design unit for the first PC it proceeds to the next. Pipelining overlaps the execution of multiple instructions within a functional unit, much like an assembly line overlaps the steps in the construction of a product [3].

In the processor there are different units, which are listed as follows:

- Fetch unit
- Decode unit
- Execute unit
- Write unit

Pipelining is controlled by the clock whose period is such that the fetching, decoding, executing and write steps of each instruction can be completed in one clock cycle [1].

The following Fig: 2 show the process of pipelining mechanism in a processor with clock cycle.

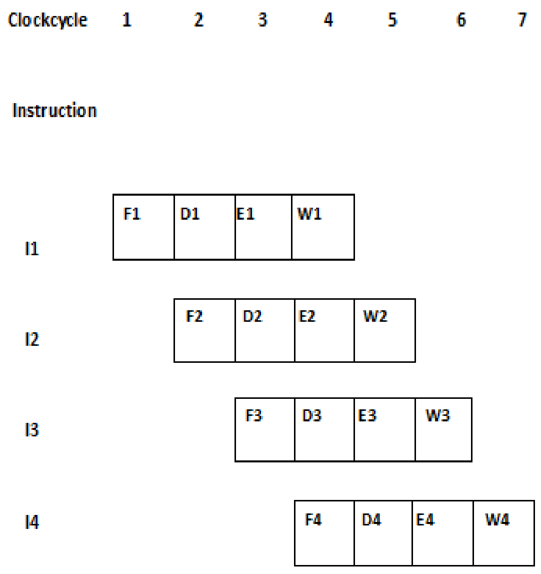

Fig: 2 Pipelined Instruction executions in four steps

3. Superscalar operation

Before the invention of the superscalar operation the execution of the instructions was done in sequential order which consumed a lot of time and ultimately resulted in the speed of the processor which was very slow in the execution process.

The following Fig: 3 show the execution of the instructions in without using the superscalar mechanism.

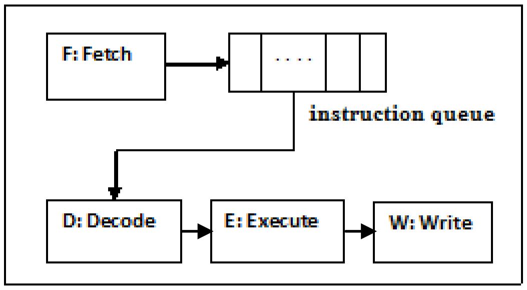

Fig: 3 normal execution of instruction

Superscalar processors include multiple functional units, such as arithmetic logic units and floating-point units. This enables the processor to exploit instruction-level parallelism (ILP), executing several independent instructions concurrently [3].

The key concept of superscalar operation os using of two separate execution units due to which the time of execution is fast than the normal pilelining processor. There buffers are used to store the instruction from the decode unit and the execution temperorily before they reach the next unit of processing mechanism.

Under the superscalar operation there are the following process [1].

- Out-of-order Execution.
- Execution Completion
- Dispatch Operation

Pipeline makes it possible to execute instruction concurrently [1]. Instructions are present in the pipelining at the same time, but they are in different stages of their execution [1]. While one instruction is performing an ALU operation, another instruction is being decoded and yet another is being fetched from the memory [1].

The front end of the processor is responsible for fetching instructions from memory and supplying them to the issue stage [4]. First the instruction is being fetched from the memory and in the instruction fetch block show the function of the fetch unit. It is followed by the instruction queue in which the instructions are stored in a queue waiting to be decoded. Then buffer are used to store the decoded instructions temporarily followed by two execution units namely,

- Floating point unit
- Integer point unit

The following Fig: 4 show the block diagram of the Superscalar property using the processor with two execution units. This gives the mechanism of the superscalar operation in a processor.

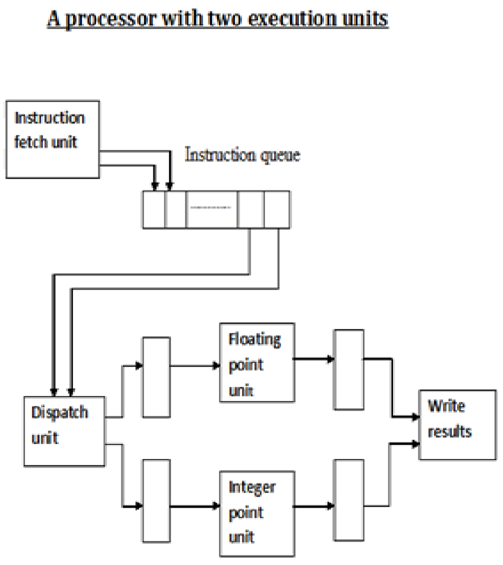

Fig: 4 Superscalar operations

By using separate execution improves the speed of the processor. Calculation of the instruction per cycle throughput (IPC) of superscalar processors [4] is very important to know the speed of the processor. High performance superscalar processors dynamically predict branches and execute instructions along the predicted control flow path [5]. Thus by using this mechanism the speed and the potential of the processor were increased.

4. Nurture IDR Segmentation and Multiple Instruction Queues in Superscalar Pipelining Processor

In Nurture IDR segmentation and multiple instruction queues in superscalar pipelining processor, we have introduced a new unit named as the Identifier unit. When the instruction is fetched from the memory by the fetch unit it enters into the identifier unit which identifies the type of the instruction and separates the instructions by creating individual segments of executions. Once the instruction type is identified it enters into its corresponding instruction queue then followed by the decoding unit and the executing unit. When the execution is completed it reaches into the write unit.

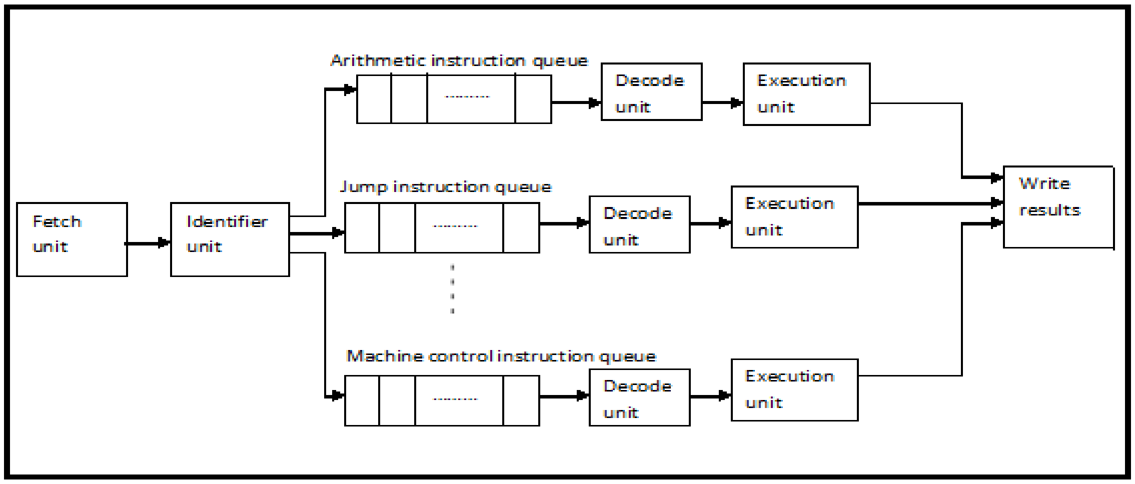

Fig: 5 Proposed Nurture IDR segmentation and multiple instruction queues

Identifier unit can be considered as the heart of the entire Nurture IDR segmentation and multiple instruction queues in superscalar pipelining processor shown in Fig: 5. Because when it separates the instruction and sends it corresponding instruction queue the time consumed is very less and ultimately the speed of the processor increases. While an instruction of one type is being processed, the next instruction is fetched and identified and separated and so on. By doing this the performances of the processor can be definitely improved. In Nurture IDR segmentation and multiple instruction queues, multiple instruction queues are used here to store different types of machine instructions in each queue. So lot of instructions are fetched and identified and stored in the corresponding queues.

The number of queue depends on the number of types of machine instruction. Each queue sends the instruction to the corresponding decode and execution units.

4.1 Fetch unit

Instruction Fetch unit fetch the instruction from the memory [1].Every fetch operation sends one instruction to the Identifier unit through directly transfer. Only after sending an instruction to the identifier unit the fetch unit will fetch the next instruction. Fetch unit will fetch the instruction one by one from the memory. If pre-fetching is not used, the fetching and the decoding widths will be equal; the instruction fetcher has the responsibility of determining the new instruction counter [6]. Fetch throughput by addressing can be improved by three factors: fetch efficiency, by partitioning the fetch unit among threads; fetch effectiveness, by improving the quality of the instructions fetched; and fetch availability; by eliminating conditions that block the fetch unit [7].Conventional instruction fetch mechanisms fetch contiguous blocks of instructions in each cycle. They are difficult to scale since taken branches make it hard to increase the size of these blocks beyond eight instructions. The wider the fetch unit, the more likely it is that fetch slots will be wasted because of discontinuities in the instruction stream [8].Only when the fetch unit functions without any interruption the instruction will be transferred to next unit through which the speed of the processor can be maintained. Here in this Nurture IDR segmentation and multiple instruction queues concept after fetching the instructions are forwarded to the identifier unit where the type is identified and the stored in corresponding instruction queues. If the speed of the fetch unit is high obviously the speed of the processor will also be high.

4.2 Identifier unit

The identifier unit is the main important unit in which major operations takes place. In this proposal we have tried to throw light on the idea of segmenting the machine instructions before decoding by using the Identifier unit. Segmentation takes place on the bases of the types of the instructions.

The list of few instructions types commonly used are shown as follows:

- Arithmetic instructions
- Logical instructions
- Jump instructions
- Data transfer instructions
- Boolean variable instructions
- Machine control instructions

These are most common types of instructions used in the programming. The identifier unit identifies the instructions based on the above types and sends it to the corresponding instruction queue and decode and the execute unit finally to the write unit store the results. The identifier unit plays the important role in Nurture IDR segmentation and multiple instruction queues in superscalar pipelining processor. The process of segmentation takes place in the identifier unit. It must identify the type of the instruction and send in the respective type of the instruction queue by which we are increasing the speed and this type of operation can store a large number of instructions in the instruction queues because there are separate queue for each type of instructions.

4.3 Multiple Instruction Queues

In Nurture IDR segmentation and multiple instruction queues in superscalar pipelining processor we have implemented multiple instruction queues. The Identifier unit indentifies the type of the instruction and stores it in the corresponding instruction queue. This give raise the concept of multiple instructions queues in the superscalar pipelining processor so large number of instructions can be stored in the processor at a time. Hence the processor witness high performances rate and an increase in the speed in made possible.

4.4 Decode unit

Decoder unit is used to convert high level language to machine understandable language. Here we have multiple decoders. For each type of instruction there are separate decoding unit. Instructions from the instruction queues are forwarded to the Decode unit based on the instruction types.

The segmentation of the instructions is done before the decoding. The decoder module decodes the VLIW passed by the memory pre fetch module and generates the control signals and register addresses for all the seven instructions forming the Very Long Instruction Word (VLIW) [9]. Thermometer-to-binary decoder can be implemented by various approaches, e.g., a ROM, Wallace-tre (oronescounter), multiplexer-based decoder, fat-tree decoder and logic-based decoder [10]. The main concept here is we are identifying the type of the instruction before the decoding process.

4.5 Execute unit

The execute stage performs ALU operations[11].The complexity of modern processors [12] has made the task of calculating or even bounding the execution time of a sequence of operations very difficult [13].The units in the execution engine are pipelined, though an EU can only issue one instruction per cycle[14].In Nurture IDR segmentation and multiple instruction queues in superscalar pipelining processor we have implemented multiple execution units, which have separate execution unit for each type of the instruction. The execution is done by the ALU individually and finally forwarded to the Write unit. The addition of execution units of the same type in order to leverage instruction level parallelism [15] .

4.6 Write unit

After the execution the instructions the results of the instructions are stored by the write unit. The write unit will store the results in the destination location [1]. Execution unit will send the executed results directly to the Write unit to process the write operation in the specified location. The process taking place in this unit is the final stage in Nurture IDR segmentation and multiple instruction queues in superscalar pipelining processor. This model is the advanced technique to improve the speed of the processor.

5. Conclusion

This paper has ultimately focused on the goal of increasing the speed of the pipelining processor mechanism and segmentation of instructions before decoding. The implementation of the identifier unit is believed to have an optimistic impact on the speed of the processor. The concurrent execution of instruction can have tremendous improvement. The implementation of Nurture IDR segmentation and multiple instruction queues in Superscalar pipelining processor will have a promising enhancement in the speed therefore reducing the execution time of the present pipelining processor. The implementation of multiple instruction queue increase the temporary storage so many instructions can be stored concurrently.

References

[1] Carl Hamacher, Zvonko Vranesic, Safwat Zaky. Computer Architecture , Mc Graw Hill, 2002.
[2] J.A. Bergstra and C.A. Middelburg, "Maurer Computers for Pipelined Instruction Processing", 1998.
[3] Chris Stolte, Robert Bosch, Pat Hanrahan, and Mendel Rosenblum , "Visualizing Application Behavior on Superscalar Processors", Stanford University.
[4] Tarek M. Taha D. Scott Wills, "An Instruction Throughput Model of Superscalar Processors" , 2003.
[5] Walid J. Ghandour June, "Dynamic Control Independence Predictor for Speculative Multithreading Processors" , 2009.
[6] Mojtaba Shojaei, Bahman Javadi*, Mohammad Kazem Akbari, Farnaz Irannejad ,"Designing and Optimizing the Fetch Unit for a RISC Core".
[7] Dean M. Tullsen, Susan J. Eggers, Joel S. Emery, Henry M. Levy,Jack L. Lo, and Rebecca L. Stammy, "Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor". Santa Margherita Ligure, Italy, June, 1995.
[8] Paramjit Oberoi and Gurindar Sohi, "Out-of-Order Instruction Fetch using Multiple Sequencers" ,Computer Sciences Department, University of Wisconsin - Madison.
[9] R. SESHASAYANAN, DR.S.K. SRIVATSA, "A novel architecture for VLIW processor", 2007.
[10] E. Sall, M. Vesterbacka, and K. O. Andersson, "A study of digital decoders in flash analog-to-digital converters," in Proceedings of IEEE International Symposium on Circuits and Systems, vol. 1, pp. 129–132, May 2004.
[11] Ben Lickly,Isaac Liu,Sungjun Kim, Hiren D. Patel,Stephen A. Edwards,Edward A. Lee, "Predictable Programming on a Precision Timed Architecture" ,2008.
[12] J. L. Henessey and D. J. Patterson, "Computer Architecture: A Quantitative Approach". Morgan Kaufmann Publishers, third edition, 2003.
[13] C. Ferdinand, R. Heckmann, and et. al. " Reliable and precise WCET determination for a real-life processor", International Conference on Embedded Software Oct. 2001.
[14] "WiDGET: Wisconsin Decoupled Grid Execution", Tiles June 2010.
[15] H. Peter Hofstee , " Power Efficient Processor Architecture and The Cell Processor ", Proceedings of the 11th Int'l Symposium on High-Performance Computer Architecture 2005.

J.Nandini Meeraa pursuing bachelor's degree in computer science and engineering third year at Narasu's Sarathy Institute of Technology, Salem. Approved by All India Council for Technical Education, New Delhi (AICTE) and is affiliated to Anna University of Technology, Coimbatore. I am a member of CSI computer society. My current research interest is on creating evolution in the speed of the processor.
.

N.Indhuja pursuing bachelor's degree in computer science and engineering third year at Narasu's Sarathy Institute of Technology, Salem. Approved by All India Council for Technical Education, New Delhi (AICTE) and is affiliated to Anna University of Technology, Coimbatore. I am a member of CSI computer society. My current research interest is on creating evolution in the speed of the processor.

S.Devi Abirami pursuing bachelor's degree in computer science and engineering third year at Narasu's Sarathy Institute of Technology, Salem. Approved by All India Council for Technical Education, New Delhi (AICTE) and is affiliated to Anna University of Technology, Coimbatore. I am a member of CSI computer society. My current research interest is on creating evolution in the speed of the processor.

K.RathinaKumar, Working as a lecturer at Narasu's Sarathy Institute of Technology, Salem in the Department of Electronics and Communication Engineering. Approved by All India Council for Technical Education, New Delhi (AICTE) and is affiliated to Anna University of Technology, Coimbatore.; published paper in the international journal in

the title of " Efficient method for escalating the performance of programmable router for network on chip by lightweight circuit switched approach" in International Journal of Communication ,Computation and Innovation[IJCSI] of volume 1, issue 2(Jan-July 2011) of ISSN 2229-6808; published paper in the international journal in the title of "Multi Machine power system stabilizer design using particle swarm optimization technique" in International Journal of Communication ,Computation and Innovation[IJCSI] of volume 1, issue 2(Jan-July 2011) of ISSN 2229-6808; Presented a paper in National level conference on significant challenges of smart antennas in ADHOC networks at Idhaya Engineering College on 26th March 2011 conducted by department of CSE, ECE & IT in the title of National Conference on Advanced Computing and Communication Systems; Presented a paper in International level conference on channel noise cancellation using blind adaptive equalization at SSM college of engineering between September 21-23, 2011 conducted by Department of ECE in the title of International Conference on Computer Communication & Signal Processing (IC^{3}SP)-2011;