© 2016 IJSRSET | Volume 2 | Issue 6 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section : Engineering and Technology

Dynamic Dispatch Scheduling for High Performance VLIW Architecture

R. Ramesh Babu[1], **Dr. Sachin Saxena**[2]

[1]Research Scholar, Department of ECE, Sunrise University, Alwar, Rajasthan, India

[2]Supervisor, Department of ECE, Sunrise University, Alwar, Rajasthan, India

ABSTRACT

VLIW architecture has become widespread due to the combined benefits of simple hardware and compiler-extracted instruction-level parallelism. However, the VLIW instruction set architecture and its hardware implementation are tightly coupled, a novel simultaneous multithreading VLIW architecture with dynamic dispatch mechanism to address the challenge of the underutilization of computing resources. The SMT technology exploits the unused instruction slots by converting the thread level parallelism to the instruction-level parallelism, improving the efficiency. On the availability of the corresponding functional units. With the dynamic dispatch mechanism, the issues instructions to functional unit at run-time rather than at compile-time, such that the issue conflicts among multiple threads are reduced significantly. The new VLIW architecture shows that it can effectively increase the processor throughput and improve the performance.

Keywords : VLIW, ILP, Reorder Buffer, Dynamic Dispatch

I. INTRODUCTION

With the advent of RISC architectures, the x86 is now recognized as a deficient instruction set. Instruction set compatibility is at the heart of the desktop microprocessor market. Because the application programs that end users purchase are delivered in binary (directly executable by the microprocessor) form, the end users desire to protect their software investments creates tremendous instruction-set inertia. RISC chips use a rather small number of relatively simple, fixed-length instructions, always 32 bits long. Although this wastes some memory by making programs bigger, the instructions are easier and faster to execute. Because they have to deal with fewer types of instructions, RISC chips require fewer transistors than comparable CISC chips and generally deliver higher performance at similar clock speeds, even though they may have to execute more of their shorter instructions to accomplish a given function.

Since the early days, asynchronous circuits have been used in many interesting applications. The results show that asynchronous circuits have advantages of low power consumption and high performance. In the embedded systems that are sensitive to power dissipation, there is a problem for us to solve that how to make our processors have lower power consumption without performance loss.

VLIW instructions are necessarily longer than RISC or CISC instructions, thus the name Very long instruction word or VLIW refers to a processor architecture designed to take advantage of instruction level parallelism (ILP). Whereas conventional processors mostly only allow programs that specify instructions to be executed one after another, a VLIW processor allows programs that can explicitly specify instructions to be executed at the same time (i.e. in *parallel*). This type of processor architecture is intended to allow higher performance without the inherent complexity of some other approaches. In VLIW machine, instructions fetched by processor in a given cycle are to be performed by several functional units. It is important to dispatch these instructions to their destinations correctly. The design of the dispatch unit in these processors is a significant impact of the implementation of a sub system. Rapidly and correctly dispatching the instructions is the necessity to prevent the performance from the bottleneck.

Very Long Instruction Word (VLIW) processors have wide acceptance in the embedded domain due to hardware simplicity, low cost and low power consumption. To exploit high Instruction Level Parallelism (ILP), VLIWs need to be designed with a significant issue width. However, the centralized Register File (RF), with all the Functional Units (FUs) connected to it, becomes a bottleneck because of an increase in RF delay, power consumption and area. Clustered VLIW architectures have multiple RFs and cluster the FUs according to the RFs they are connected to. Many VLIWs have been designed using the clustered approach. Some applications do not take advantage of the high issue width available in a VLIW processor and the processor is heavily underutilized. In the context of VLIW architectures, processor underutilization can be characterized in terms of vertical and horizontal waste. Vertical wastes are the cycles where no operations are issued at all. Horizontal waste is the underutilization of the issue width of the processor, i.e. the number of operations issued in a cycle is less than the issue width. Several multithreading techniques have been proposed to reduce the vertical and horizontal waste in the processor.

Traditional approaches to improving performance in processor architectures include breaking up instructions into sub-steps so that instructions can be executed partially at the same time (pipelining), dispatching individual instructions to be executed completely independently in different parts of the processor (superscalar architectures), and even executing instructions in an order different from the program (out-of-order execution). These approaches all involve increased hardware complexity (higher cost, larger circuits, higher power consumption) because the processor must intrinsically make all of the decisions internally for these approaches to work. The VLIW approach, by contrast, depends on the programs themselves providing all the decisions regarding which instructions are to be executed simultaneously and how conflicts are to be resolved. As a practical matter this means that the compiler software (the software used to create the final programs) becomes much more complex, but the hardware is simpler than many other approaches to parallelism. As is the case with any novel architectural approach, the concept is only as useful as code generation makes it. However, these optimized capabilities are useless unless compilers are able to spot relevant source code constructs and generate target code that duly utilizes the CPU's advanced offerings. Therefore, programmers must be able to express their algorithms in a manner that makes the compiler's task easier.

As SMT exploits some of the unused instruction slots by converting thread-level parallelism (TLP) to ILP, a processor with SMT can issue multiple instructions from multiple threads each cycle. Therefore, SMT improves the processor throughput, raises the functional unit utilization, and exploits maximum ILP by issuing as many instructions as possible from multiple threads in any given cycle.

Instruction Level Parallelism

Very Long Instruction Word (VLIW) processors have wide acceptance in the embedded domain due to hardware simplicity, low cost and low power consumption. To exploit high Instruction Level Parallelism (ILP), VLIWs need to be designed with a significant issue width. However, the centralized Register File (RF), with all the Functional Units (FUs) connected to it, becomes a bottleneck because of an increase in RF delay, power consumption and area

Clustered VLIW architectures have multiple RFs and cluster the FUs according to the RFs they are connected to. Many VLIWs have been designed using the clustered approach. Some applications do not take advantage of the high issue width available in a VLIW processor and the processor is heavily underutilized. In the context of VLIW architectures, processor underutilization can be characterized in terms of vertical and horizontal waste. Vertical wastes are the cycles where no operations are issued at all. Horizontal waste is the underutilization of the issue width of the processor, i.e. the number of operations issued in a cycle is less than the issue width. Several multithreading techniques have been proposed to reduce the vertical and horizontal waste in the processor.

Architectures exploiting instruction-level parallelism (ILP) at compile time, such as Very Long instruction word (VLIW) and transport triggered architecture (TTA), may satisfy the requirements. They can be further enhanced by using asynchronous circuits to significantly reduce power consumption. As such, we are interested in asynchronous processors with architectures exploiting ILP at compile time. However, most of the current asynchronous processors are based on RISC-like architectures. When designing asynchronous VLIW or TTA processors, the distribution of control introduces some serious problems, and errors may occur because of the variable latencies of operations.

Instruction-Level Parallelism (ILP) is a measure of how many operations in a computer program can be executed concurrently. An ILP processor has multiple function units that can be engaged simultaneously executing multiple operations.

In Figure1showing a simple five-stage processor pipeline in the multiple-issue case on the right, there can be two operations in the EX stage in a single cycle. Noteworthy, in the same cycle multiple register file accesses can occur (write backs in the WB stage and operand reads in the ID stage), requiring a multi-ported register file.

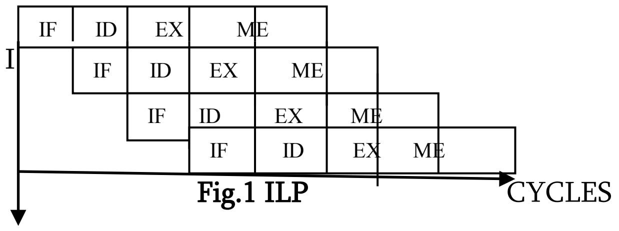


Fig.1 ILP

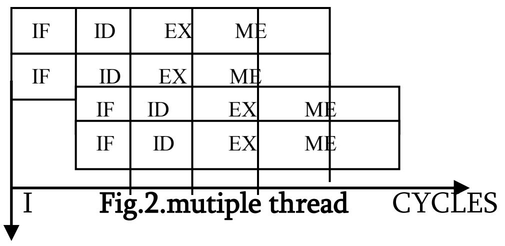


Fig.2.mutiple thread

Previous studies of ILP limits indicate availability of *potentially* high operation concurrency in (media) applications, spanning the range of a few tens up to hundreds of independent operations. For example, Table II shows Potential ILP rates for a basic block instruction scheduling, program wide instruction scheduling and program-wide scheduling combined with aggressive code optimizations.

Reorder buffer

Additional set of hardware registers called reorder buffers (ROBs). ROBs stores results of instructions (in shadow) that have completed but not yet committed. Instructions enter ROB out of order and instruction leaves ROB in order. Results of an instruction become visible externally when it leaves ROB and force them to complete in order. Elimination of store buffers and replacing them by ROBs in VLIW architecture. Exceptions are masked until instructions commits.

Instruction Format

The Instruction Set of processor consists of 16-bit instructions and 32-bit instructions. Most frequently used instructions, such as addition, subtractions, and/or, etc, have both 16-bit 32-bit instruction formats.
The Processor instruction set picks up the parallel information from the instruction type and the register

file assignment field. According to the register file assignment field, the instructions are grouped into two slots, slot x and slot y, and each slot consists of up to three instructions. The slot x and slot y can be executed concurrently, and the instructions in the same slot can be dispatched in parallel.

Instruction Dispatch

When multiple threads contend for one functional unit, an issue conflict on that functional unit occurs. To quantify the issue conflicts, the issue conflict rate is introduced to indicate the most wanted functional unit for both threads in a given period. Equation 1 calculates the issue conflict rate, in which issue conflict cycle is the cycle when an issue conflict occurs.

Issue Conflict Rate = Issue Conflict Cycles / Total Cycles (1)

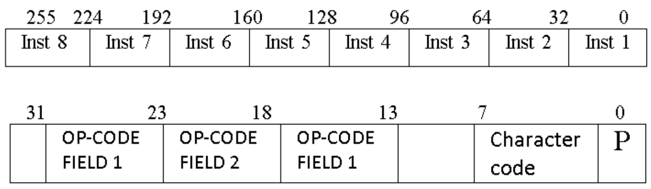


Fig.3 : The block diagram of instruction dispatch unit

Conventional VLIW assign the instructions to functional unit at compile-time with the functional unit assignment field in the operation code.

Destination Identification

The instructions of an execution packet may be performed by different functional units. The dispatch unit checks the aims of them and delivers them to the proper places, where pre-process and decode instructions and then perform the operations.

Pipeline Partially Discontinued

In the case of more than one execution packet in the fetch packet, the dispatch unit needs some more pipeline cycle to finish dispatching these eight instructions. Some of the stages of the pipeline have to be stopped, such as the instruction fetch unit; while some of other stages, ALUs, ought to run normally. When the final execution packet of the fetch packet has been delivered, the pipeline continues. It is indispensable for the DSP to run correctly; otherwise undefined erroneous result would be produced.

Branch Consideration

It is common in the program to switch the section. When the fetch packet of the destination of the branch instruction comes to the instruction dispatch unit, it is definite to begin the dispatch and delivery of the eight instructions immediately, abandoning the processing of the previous fetch packet.

Instruction Pre-Processing

According to the statistics of the Processor performance, it is the instruction decoding and preparing the operands that take up some long time in a cycle, which is the critical path delay in design. Therefore the instruction pre-processing is very helpful to improve the clock rate, as it interprets the type of operation and the source of operands. Thus, to sum up the points indicated above, it is the complexity of the instruction dispatch unit that makes the design of the logic circuitry very complicated. At the same time, the area and power consumption are serious disadvantages of the circuit unit.

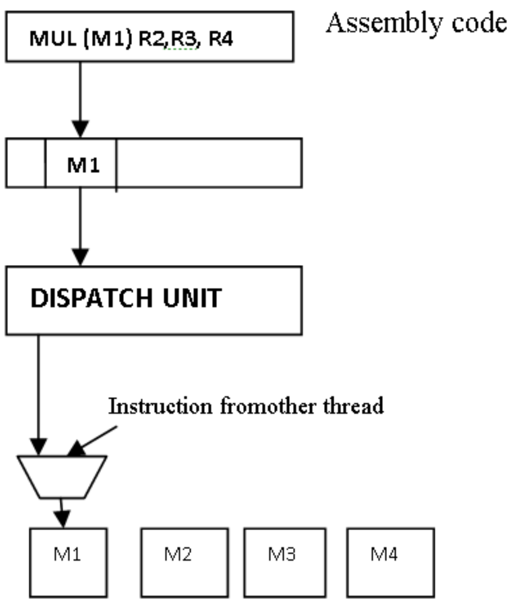


Fig.4 (a). Static dispatch mechanism

Figure.4 (a) according to the functional unit assignment field in the instruction code, the dispatch unit issues the instruction to the M1 unit. This can be called static dispatch mechanism. Though there are multiple M units, M1 to M4, which can execute this multiply instruction, the hardware just issues the instruction to M1 unit which is designated by the complier. If M1 unit has been taken up by the higher priority thread, the issue conflict occurs and the instruction is not issued until M1 unit is available. With dynamic dispatch mechanism, the pre-decode logic in the dispatch unit decodes the instruction type from the instruction code.

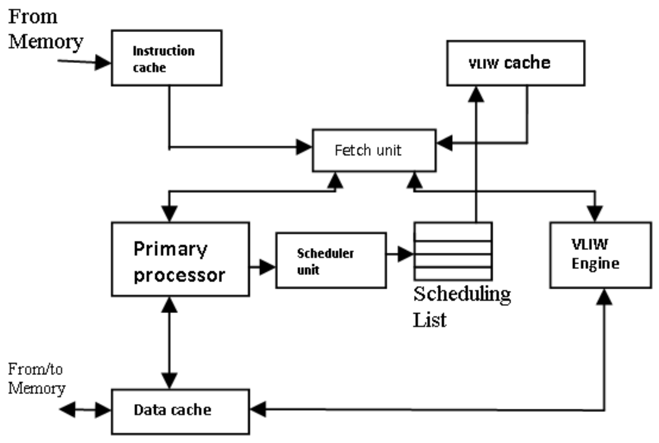


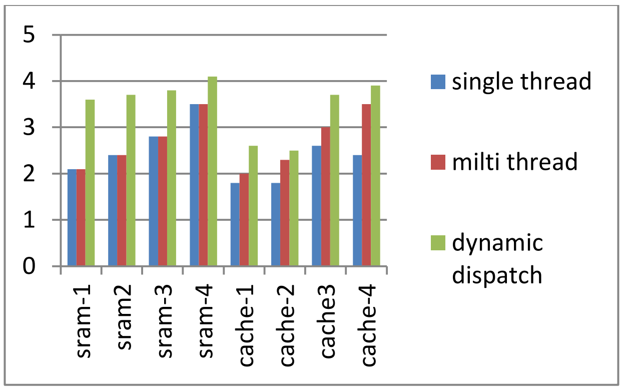
Fig.4(b). Dynamic dispatch mechanism

Figure.4 (b) shows a block diagram of the VLIW architecture. In the VLIW architecture, the scheduler engine fetches instructions from the instruction cache and executes them first using a simple pipelined processor the primary processor. In addition, its scheduler unit dynamically schedules the trace produced during this execution into VLIW instructions, placing them as blocks of VLIW instructions in the VLIW cache. If the same code is executed again, it is then fetched by the VLIW engine from this cache and executed in a VLIW fashion. In the VLIW architecture, the scheduler engine provides object code compatibility, and the VLIW engine provides VLIW performance and simplicity. Executing code in two distinct modes, one sequential and one parallel, results in four positive characteristics. First, code compatibility between different machine

Dynamic dispatch mechanism is also beneficial to the code density. With static dispatch mechanism, the instruction mapped to different functional units requires different instruction codes, even though the instruction has the same function. The dynamic dispatch unit issues instructions to functional units according to the instruction type, so that an instruction has a uniform instruction code. This orthogonal instruction set maintains a compact code density.

Evaluation of work

To evaluate the performance of our approach, we evaluate the behaviour of the prototype processor under different configurations: with on-chip memories or caches, with static or dynamic dispatch mechanisms, with single thread or dual thread, and so on.



x-axis-program,y-axis-IPC

Fig.5. Processor throughput

The generated programs (prog1~prog4) with different levels of instruction parallelism are evaluated for various configurations: with on-chip memories or caches, with static or dynamic dispatch mechanisms. Compared to the single thread model, all dual-thread models show substantial gains in IPC. If the programs are evaluated without cache, the lower the instruction parallelism of the program is, the larger enhancement can be gained from the simultaneous multithreading.

The processor throughput (IPC) for our approach in comparison with the single-thread model and the BMT model. The Block Multi-Threading (BMT) model is a dual-thread multithreading model without splitting the VLIW execution packets. Compared to the single-thread model, all dual-thread models show substantial gains in IPC. If the programs are evaluated without cache, the lower the instruction parallelism of the program is, the larger enhancement can be gained from the simultaneous multithreading. Under the configuration with cache, because the multi threading can hide the caches miss latencies, the gain in IPC of our approaches more significant. As seen in Fig.5, with respect to the single-thread model, the average processor throughputs improvement gained by our approach of the generated programs is about 52%, and about 22% with respect to the BMT model.

II. CONCLUSION

The dispatch unit of the VLIW dynamically dispatches instructions to the functional units at run-time rather than at compile-time, such that the issue conflicts among multiple threads are reduced. Instruction dispatch unit for design based on VLIW architecture. The unit recognizes the proper instructions to allocate in every cycle, and then dispatches them to the accurate functional units.

The proposed dual-thread VLIW processor model based on the Weld architecture paradigm tolerates latencies by creating a speculative thread from a single application and running it with the no speculative main thread simultaneously. The compiler decides where and when to create the speculative thread that can be both control and memory speculative at the time of thread creation.

III. REFERENCES

[1] Fisher Joseph A(1983)."Very Long Instruction Word architecture and the ELI-512" Proceedings of the 10th annual international symposium on Computer architecture. International Symposium on Computer Architecture. New York, NY, USA:AMC.140–150.DOI:10.1145/800046. 801649 .ISBN0-89791-101-6 Retrieved 2009-04-27.

[2] Manoj Gupta, Ferm´ın S´anchez,Josep Llosa 978-1-4244-6443-2/10/$26.00 ©2010 IEEE, "A Low Cost Split-Issue Technique to Improve Performance of SMT Clustered VLIW Processors".

[3] Zheng Shen, Hu He,Yihe Suna at the National Natural Science Foundation of China (NSFC) under grant No.60236020 IEEE 2009 12th Euromicro Conference on Digital System Design / Architectures, Methods and Tools "Simultaneous Multithreading VLIW DSP Architecture with Dynamic Dispatch Mechanism".

[4] J.Poornima1,G.V.Ganesh,G.Venkata Rao,S.Daya Sagar."Design and implementation of advanced 32bit pipelined RISC processor for multipurpose applications" International Journal of VLSI & Signal Processing Applications, Vol. 2,Issue2,April 2012, ISSN 2231-3133, (153-159)

[5] Bharath Iyer, Sadagopan Srinivasan, and Bruce Jacob, Proceedings of the 31st Annual International Symposium on Computer Architecture (ISCA'04) 1063-6897/04 ©2004 IEEE, "Extended Split-Issue: Enabling Flexibility in the Hardware Implementation of NUAL VLIW DSPs".

[6] Dongrui She, Yifan He, Bart Mesman, Henk Corporaal,978-3-9810801-8-6/DATE12/2012 EDAA,"Scheduling for Register File Energy Minimization in Explicit Datapath Architectures,"

[7] Yong Li, Zhi-ying Wang and Kui Dai, Seventh International Conference on Computer and Information Technology, 0-7695-2983-6/07 © 2007 IEEE DOI 10.1109/CIT.2007.53, "A Low-Power Application Specific Instruction Set Processor Using Asynchronous Function Units".

[8] Qingwei Zheng, Zhaolin Li, Jianfei Ye, Chipin Wei and Jiajia Chen, 2010 Second Pacific-Asia Conference on Circuits, Communications and System (PACCS) ©2010 IEEE,"Implementation of an Instruction Dispatch Unit Applied to Digital Signal Processors with VLIW Architecture".

[9] Mengjun Sun, Zheng Shen, Hu He, 978-1-4244-3870-9/09 ©2009 IEEE,"An Efficient Parallel Instruction Execution Method for VLIW".

[10] Ittetsu Taniguchi, Mitsuya Uchida, Hiroyuki Tomiyama, Masahiro Fukui,Praveen Raghavan and Francky Catthoor, 2011 14th Euromicro Conference on Digital System Design, 978-0-7695-4494-6/11©2011IEEE DOI 10.1109/DSD.2011.93. "An Energy Aware Design Space Exploration for VLIW AGU Model with Fine Grained Power Gating".

[11] Tien-Wei Hsieh, Pi-Chen Hsiao, Che-Yu Liao, Hsien-Ching Hsieh, Huang-Lun Lin Tay-Jyi Lin, Yuan-Hua Chu, and An-Yeu (Andy) Wu, 2008 IEEE International SoC Design Conference, "Energy-Effective Design & Implementation of an Embedded VLIW DSP".

[12] Chan-Hao Chang,DianaMarculescu Proceedings of the 2006 Emerging VLSI Technologies and Architectures (ISVLSI'06) 0-7695-2533-4/06 - 2006 IEEE "Design and Analysis of a Low Power VLIW DSP Core".

[13] Hsien-Ching Hsieh, Shui-An Wen, Che-Yu Liao, Huang-Lun Lin, Po-Han Huang,Shing-Wu Tung at 2011 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS) December 7-9, 2011 "Low Power Design and Dynamic Power Management System for VLIW DSP Subsystem" .

[14] Mostafa E. A. Ibrahim, Markus Rupp, and S. E.-D. Habib ©2009 IEEE "Performance and Power Consumption Trade-offs for a VLIW DSP".