International Journal on Future Revolution in Computer Science & Communication Engineering
Volume: 2 Issue: 6
ISSN: 2454-4248
107 – 110

Designing with RoBs for High Performance VLIW Architecture

R. Ramesh Babu, Research Scholar, Dept. of ECE, Sunrise University, Alwar, Rajasthan

Dr. Sachin Saxena, Supervisor, Department of ECE, Sunrise University, Alwar, Rajasthan

Abstract—VLIW architecture has become widespread due to the combined benefits of simple hardware and compiler extracted instruction level parallelism. The VLIW instruction set architecture and its hardware implementation is tightly coupled and a novel simultaneous multithreading VLIW architecture with dynamic dispatch mechanism which uses RoBs complex logic to maximize ILP has been proposed. Since the resulting dynamic instruction schedule of many applications seldom changes, it is reasonable to store and reuse the schedule instead of reconstructing it each time. The new VLIW architecture shows that it can effectively increase the processor efficiency which improves the performance.

Keywords—*VLIW, ILP, RoB, dynamic dispatch.*

*****

I. INTRODUCTION

With the advent of RISC architectures, the processor 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 always 32 bits long fixed-length instructions. It wastes memory for complex programs, the instructions are easier and faster to execute. Because it has to deal with some types of instructions, RISC chips require less transistors than CISC chips and get higher performance at similar clock speeds.

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 execute instructions one after another, this processor accepts programs that can explicitly specify instructions to be executed at the same time. 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.

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, 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. VLIW simply moves complexity from hardware into software.

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, dispatching individual instructions to be executed completely independently in different parts of the superscalar processor, and even executing instructions out-of-order. These approaches all involve increased hardware complexity because the processor must make all of the decisions internally for these approaches to work.

The VLIW approach depends on the programs themselves providing all the decisions regarding which instructions are to be executed simultaneously. As a practical matter this means that the compiler software becomes much more complex, but the hardware is simpler than many other approaches to parallelism. In this approach code generation makes it useful. However, these optimized capabilities are useless unless compilers are able to identify

 ISSN: 2454-4248
107 – 110

relevant source code constructs and generate target code. 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.

II. 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 ILP at compile time, such as VLIW and transport triggered architecture (TTA) may satisfy the requirements. They can be further enhanced by using asynchronous circuits to significantly reduce power consumption. In asynchronous processors with architectures exploiting ILP at compile time. 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 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 Figure.1 showing a simple five-stage processor pipeline in the multiple-issue case on the right, there can be two operations in the execution stage in a single cycle. 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.

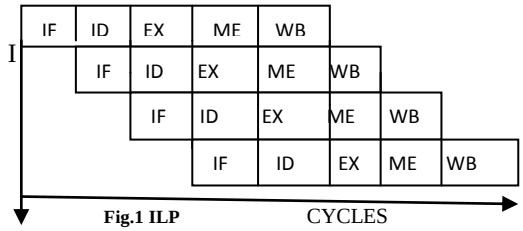

Fig.1 ILP

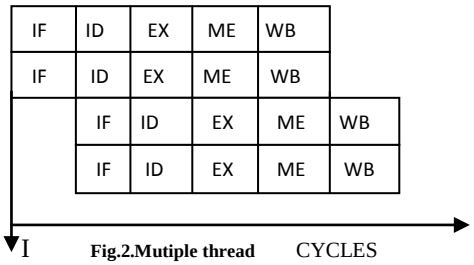

Fig.2.Mutiple thread

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 either 16-bit or 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.

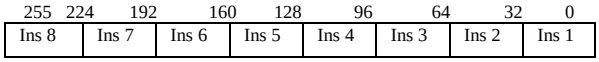

Fig.3: The block diagram of instruction format

III. DYNAMIC 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. 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.

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

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 instructions. Some of the stages of the pipeline have to be stopped, such as the

 ISSN: 2454-4248
107 – 110

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.

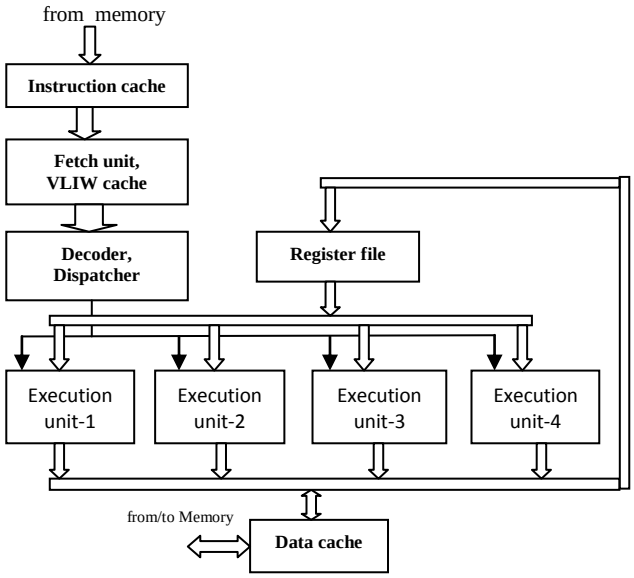

Fig.4. Dynamic dispatch mechanism

Figure.4 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 instructions in the 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.

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.

IV. PROPOSED WORK

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.

Figure.5 shows a block diagram of a VLIW processor with RoBs implementation. The implementation consists of a collection of execution that are fed operations from an instruction queue and operands from a register file and data cache.

The new VLIW processor divides the processing of instruction cycle into number of steps during this stage first it will fetch the instructions, and then the instructions are dispatched to the instruction queue until the required operands enter, the instructions remains in the queue after that the instruction is allowed to leave the queue. The instruction is moved into the appropriate execution unit. The results are placed into the RoBs. Only after all previous instructions which have to be executed before this instruction have their results written back to the register file and then this result is written back to the register file.

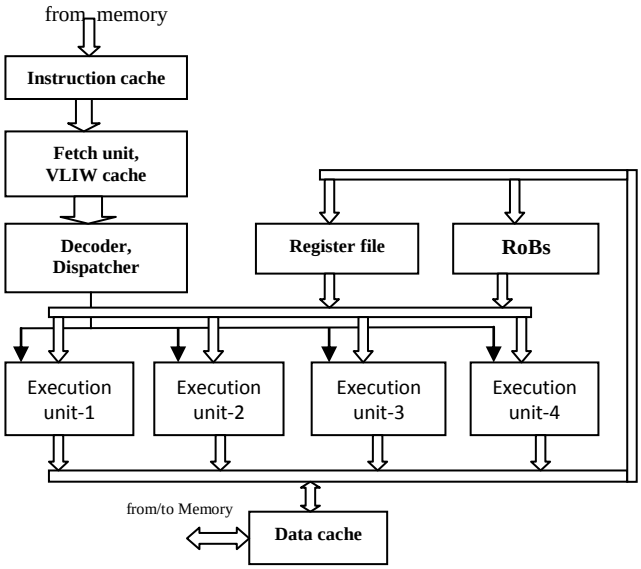

Fig.5. Dynamic dispatch mechanism with RoBs

The concept dynamic dispatch is to allow the processor to avoid a in-order of instructions pushed into the vacant slots that occur when the operands needed to perform an operation are unavailable. In the dynamic dispatch scheduling with RoBs the VLIW processor avoids the slot that occurs during the instruction dispatch of the in-order processor when the instruction is not completely ready for the processing due to missing data.

V. CONCLUSION

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

We propose dynamic VLIW processor with RoBs that saves energy when applications have regular behaviour

 ISSN: 2454-4248
107 – 110

(branch predictability and cache misses) by running in VLIW mode and maintaining the performance of a regular VLIW processor. Our proposals can benefits from the RoB logic to detect when regular code executes. When the code becomes a hotspot, it switches the back end execution mode to VLIW, saving power from the dynamic dispatch with RoBs structures not used and by turning off the functional units not required.

The advantage of VLIW processor grows as the instruction pipeline deepens and the speed difference between main memory or cache memory and the processor increases.

REFERENCES

[1] 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".

[2] 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".

[3] 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".

[4] 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".

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

[6] 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".

[7] 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" .

[8] Emre Ozer and Thomas M. Conte, IEEE transactions on parallel and distributed systems, vol. 16, no. 12, December 2005. "High-Performance and Low-Cost Dual-Thread VLIW Processor Using Weld Architecture Paradigm".