Slo-Li CHU, Geng-Siao LI, Ren-Quan LIU

Chung Yuan Christian University

Caliburn: a MIPS32 VLIW Processor with Hardware Instruction Morphing Mechanism

Abstract. *This work proposes a novel quad-issue VLIW architecture, called Caliburn, for directly executing legacy MIPS32 binary programs. To schedule and pack legacy MIPS32 binary codes on-the-fly, Caliburn has an integrated novel hardware instruction morphing mechanism that converts legacy MIPS32 binary instructions into a VLIW instruction bundles without the intervention of software compilers. The performance enhancement of Caliburn with a pipelined MIPS32 processor is evaluated. The Caliburn VLIW processor is implemented using Bluespec SystemVerilog HDL and synthesized using Synopsys Design Compiler. The experimental result reveals that the Caliburn processor achieves 3.08X speedup, and can be operated at a frequency of 425 MHz by the fabrication of TSMC 40nm technology library.*

Streszczenie. *W artykule przedstawiono propozycję nowej struktury VLIW na potrzeby wykonywania programów w architekturze MIPS32. W rozwiązaniu zastosowano technikę morfingu, w celu eliminacji programowych kompilatorów. Wykonano badania eksperymentalne na procesorze MIPS32, potwierdzające efektywność i szybkość opracowanej architektury. (**Caliburn – procesor VLIW MIPS32 ze sprzętowym mechanizmem morfingu**).*

Keywords: MIPS32 VLIW Processor, Hardware Instruction Morphing, Bluespec SystemVerilog, Binary Translation
Słowa kluczowe: procesor VLIW MIPS32, hardware'owy morfing, Bluespec SystemVerilog, translacja binarna.

Introduction

Continuous advances in semiconductor technology have made the modern processor more complex and improved instruction-level parallelism (ILP). VLIW architectures, including Intel Itanium and Transmeta Crusoe, are attractive for increasing ILP in processors without the need for sophisticated reordering and scheduling mechanisms. However, the need to re-compile applications limits the range of adoption of these architectures. Owing to the issue of the compatibility of existed binary codes, software programs must be recompiled from source codes and cannot be used to execute directly existing applications. Also, the instruction packing ratio and ILP of conventional VLIW compilers are limited by run-time status of the program, and the capabilities of VLIW processors cannot be fully exploited accordingly. This work develops a novel VLIW architecture, Caliburn, for directly executing legacy MIPS32 binary programs. Caliburn comprises quad-issue VLIW pipeline execution units and with a forwarding unit, to decode/execute whole MIPS32 integer instruction sets. To schedule and pack the legacy MIPS32 machine codes in real time, a novel hardware instruction morphing mechanism is developed to schedule/pack conventional MIPS32 binary instructions into VLIW instruction bundles without the intervention of a compiler. To elucidate the advantages of Caliburn VLIW processor, the performance of a Caliburn VLIW processor is evaluated. The Caliburn VLIW processor was implemented using Bluespec SystemVerilog and synthesized using the Synopsys Design Compiler. The experimental results indicate that Caliburn processor achieves a 3.08X performance enhancement than conventional pipelined MIPS32 processor. The working frequency of 425 MHz can be achieved under the fabrication of TSMC 40nm technology.

Related Works

To increase the instruction level parallelism (ILP) of modern microprocessors, more functional units must be used to execute multiple issued instructions concurrently. Such microprocessors are called superscalar, and can be classified as dynamic superscalar or static superscalar. Most dynamic superscalar architectures use an out-of-order scheduling mechanism and reordering buffer to analyze dependence, schedule instructions, issue instructions, and execute instructions on-the-fly. Static superscalar architectures, including VLIW architectures, can execute multiple instructions concurrently, but cannot perform dynamic scheduling or the dynamic issuing of instructions [1] [3] [5]. Therefore, VLIW architectures require specific VLIW compilers to analyze and pack instructions that can be executed concurrently in an "instruction bundle" [6]. The software that is designed for VLIW processors must be re-compiled. This requirement reduces their usefulness. Hence, many studies of binary translation and dynamic scheduling/execution of unpacking instruction streams have been published in recent years.

The Itanium [6] architecture is a 64-bit microprocessor architecture that was designed by HP and Intel. It combines the features of RISC and VLIW, and can execute an EPIC instruction set. The VLIW compiler of the Itanium processor can group independent instructions, exploit instruction level parallelism, and finally pack independent instruction pairs into a single EPIC instruction bundle. Despite the cache, the hardware complexity of Itanium is lower than that of a modern superscalar processor, such as a Pentium 4, because the Itanium compiler reorders instructions, instead of hardware scheduler. However, the major challenge associated with the Itanium processor concerns binary compatibility. Existing software must be recompiled, which limits the range of uses of the Itanium processor.

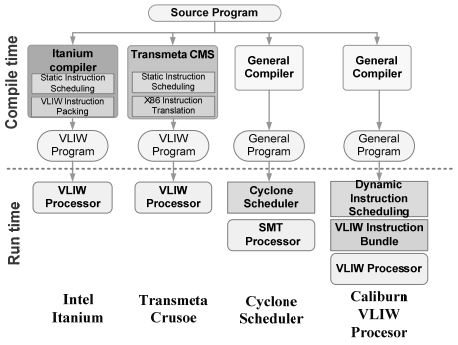

Fig.1. Execution flows of four VLIW architectures

The Cyclone scheduler [3], developed by Ernst et al., is a dynamic instruction scheduler for the multiple issues superscalar processor. By using the proposed list-based, single-pass instruction scheduling algorithm, Cyclone can schedule and reorder instructions to improve the ILP of the superscalar processor. The kernel of the Cyclone scheduler

is the time estimation mechanism, which can estimate the execution time of the instructions. If the real execution time is less than the estimated time, then scheduler will execute those instructions. In contrast, if the real execution time is larger than the estimated time, then the Cyclone scheduler inserts instructions into the replay queue.

This section considers in detail the design and implementation of the proposed Caliburn VLIW architecture. First, the architecture of the baseline MIPS32 processor is designed and the effectiveness of the integer MIPS32 instruction decoder and the five-stage pipelined processor is verified. The basic building blocks, including the RISC pipeline, arithmetic logic unit, register file, memory interface, control unit, delayed branch, and forwarding unit, are developed in this stage. Second, the quad-issue VLIW processor is designed as an extension of the baseline MIPS32 architecture. Since the basic MIPS32 processor was designed in the preceding stage, the main objective of this second stage is to form a quad-issue datapath and a dedicated forwarding unit to handle the data dependence of instructions in the quad-issue datapath. The corresponding control unit, instruction decoder, and arithmetic and logic units are extended or duplicated to meet the requirements of the quad-issue processor. The complex register file is designed for sharing among the quad-issue pipelines. Third, the proposed sophisticated Hardware Instruction Morphing (HIM) mechanism is developed.

Figure 2 shows the architecture of the proposed baseline MIPS32 processors, which comprises a five-stage pipeline. The stages are Instruction Fetch (IF), Instruction Decode and Register Fetch (ID), Execution (EX), Memory (Mem), and Write Back (WB). The first objective of designing this baseline MIPS32 processor is to verify the correctness of the integer MIPS32 instruction decoder. The implemented instructions include 79 MIPS32 integer instructions and two exception instructions. The second objective is to construct the fundamental pipelined datapath and the corresponding functional units, which are the arithmetic and logic unit (ALU), register file, memory access interface, program counter, and forwarding unit. After this baseline MIPS32 processor is designed the development steps described below can be accomplished.

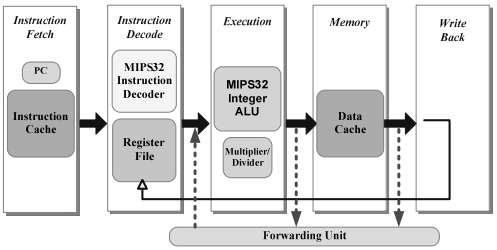

Fig.2. Organization of five-stage pipelined MIPS32 processor

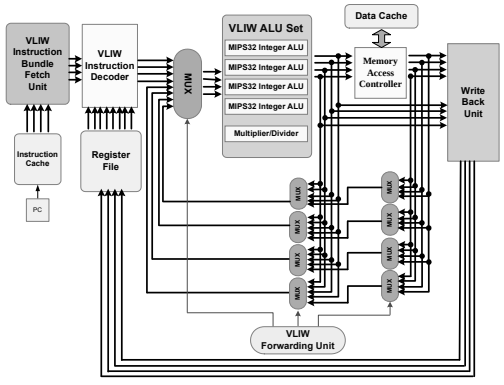

Fig.3. Datapath of quad-issue VLIW processor with HIM echanism

Based on the baseline MIPS32 processor that was described in the preceding subsection, the quad-issue MIPS32 VLIW processor can be constructed by using and extending the functional units of the baseline MIPS32 processor. Figure 3 presents the architecture of the quad-issue MIPS32 VLIW processor. To extend to quad-issue, the datapath undergoes various modifications. These include the duplication of ALU, control signals, datapaths, and the MIPS32 instruction decoder. The register file and forwarding unit must be redesigned for the VLIW datapath. First, the register file must be modified from its two-read one-write capability to eight-read four-write capability, to share access among the four datapaths. Second, the forwarding unit must be rewritten for the transfer of operands among the four datapaths when the instruction contains data dependencies.

The primary modification of the register file is the change in its accessing interface from two-read one-write capability to eight-read four-write capability. The number of registers remains 32. Since the four datapaths may read the same register at the same time, the modified read interface has to serve immediately.

Figure 4 shows the architecture of the proposed Hardware Instruction Morphing (HIM) mechanism. Unlike the dynamic scheduler in conventional out-of-order superscalar processors, the proposed HIM mechanism analyzes instructions in simply way; detects the basic block; analyzes the dependence relations of the instructions; detects the types of hazards, and packs four instructions into an instruction bundle for the Caliburn VLIW processor. Since the HIM mechanism does not need to track the entire execution flow of each instruction, or to schedule instructions between the major functional units of the superscalar processor, the only purpose of the HIM mechanism is to select four instructions and pack them into an instruction bundle for the Caliburn VLIW processor. The hardware complexity is thus dramatically reduced. Table 1 presents the types of instructions of MIPS32 ISA that utilized by the Caliburn VLIW processor.

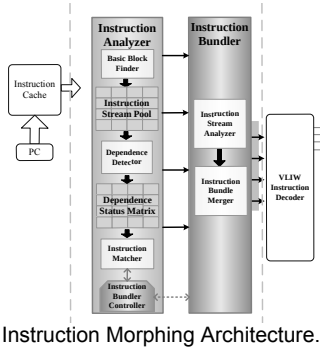

Fig.4. Hardware Instruction Morphing Architecture.

Table 1. Classification of MIPS32 instructions

	Type	Instruction Field			
		rs (r)	rt (r/w)	rd (w)	sa (r)
1	R type 1	V	V (Read)	V	
2	R type 2		V (Read)	V	V
3	Special R type 1	V			
4	Special R type 2			V	
5	Special R type 3		V (Read)	V	
6	Special R type 4	V		V	
7	I type 1	V	V (Read)		
8	I type 2	V	V (Write)		

All of the adopted MIPS32 instructions can be classified into eight types, based on their formats and features. (1) R type 1 and (2) R type 2 include most MIPS R-type instructions that use three operands and do not require an immediate value. The difference between R type 1 and R type 2 is the fields of rs (r) and sa (r) operands. Special R

type instructions can be classified into four sub-types based on their operands. I type 1 and I type 2 are MIPS I-type instructions. They have the same operand format but with different meanings. This classification of instructions substantially reduces the computation time that must be made by the Instruction Matcher to perform the analysis.

Architecture of Caliburn VLIW Processor

Figure 5 presents the datapath of the Caliburn VLIW processor. To increase the number of instructions that can be concurrently executed, Caliburn is composed of four functional units, and a VLIW register file that can support eight concurrently read and four concurrently write. Since the number of concurrently executed instructions in the VLIW datapath is increased, the dependence relations of executed instructions increase. In the single-issue, five-stage pipelined processor, stalling instructions to prevent data dependence can solve this problem. However, in a four-issue VLIW processor, Caliburn, the stall penalty is serious if data-dependent instructions execute frequently. To solve this problem, the dedicated forwarding datapath for Caliburn VLIW processor is developed. The execution flow of the proposed Caliburn VLIW processor is as follows. Firstly, the MIPS32 binary instructions are fetched from the Instruction Cache, according to the starting address stored in PC. Then, the fetched instructions are fed into the Hardware Instruction Morphing stage of the Caliburn VLIW processor, to be analyzed, scheduled, and packed into a VLIW instruction bundle on-the-fly. The packed VLIW instruction bundle is then decoded using an Instruction Decoder. Each execution step of the Caliburn VLIW processor is much simpler than conventional dynamic superscalar processor, such as an Intel Pentium 4, because functional units of Caliburn processor operate in a fixed orientation. Conventional dynamic superscalar processors that adopt Tomasulo's algorithm and a reordering buffer to schedule instruction stream, their execution flow may change every cycle because the completion time is variant and the commit order may change.

Experiment Results

The proposed Caliburn VLIW processor is implemented using Bluespec SystemVerilog [2], which is a new electronic system-level (ESL) hardware description language for unifying the hardware design flow from high-level modeling to Verilog chip design. The benchmark program for evaluating the speedup of Caliburn VLIW processor versus baseline MIPS32 processor is compiled by the MIPS SDE Lite [4] compiler. Figure 6 plots the experimental results concerning speedups with variant sizes of basic block. Since the Caliburn processor is a quad-issue VLIW architecture, the speedup is less than 1.0 if the basic block size is less than four. The Hardware Instruction Morphing (HIM) that is proposed in Section 4 uses the branch instruction as the boundary, so the VLIW instruction bundle will be filled with instructions if the basic block size is less than four. A basic block comprises two branch instructions, and the number of packed VLIW instruction bundles exceeds three. Therefore, the speedup of 2.51X is achieved when the basic block size exceeds 16. The initial cost of the basic block can be neglected. Finally, when the basic block size exceeds 32, a speedup of 3.08X can be achieved, indicating that the quad-issue VLIW processor can actually issue, execute, and commit instructions with 3.08 time performance improvement. The Verilog implementation of the Caliburn VLIW processor, generated from the Bluespec SystemVerilog design, is performed by using the Synopsys Design Compiler with TSMC 40nm technology library. The clock period is less than 2.35 ns, and the Caliburn VLIW processor can achieve a working frequency of 425 MHz. The chip area is 147887 μm^2 and the corresponding power consumption is 23.5 mW.

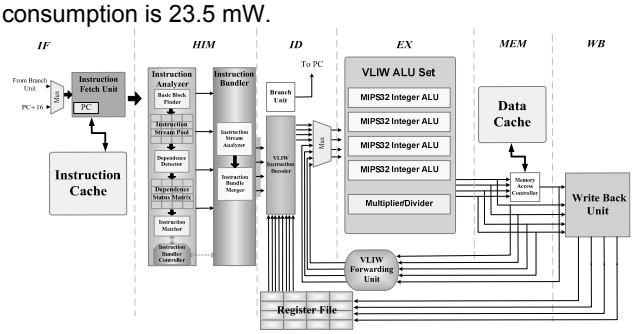

Fig. 5. Architecture of Caliburn VLIW processor.

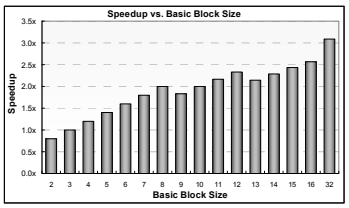

Fig.6. Speedup versus Basic Block Size.

Conclusions

Conventional VLIW processors can achieve a high ILP without a complex instruction reordering mechanism, but its use is limited by the problem of binary codes compatibility. Software programs have to be recompiled and the hardware resource of VLIW architectures must be fully utilized to obtain better performance. This work proposes the Caliburn VLIW architecture for executing legacy MIPS32 binary programs without recompiling. A novel hardware instruction morphing mechanism is integrated into Caliburn to pack conventional MIPS32 binary instructions into VLIW instruction bundles without any intervention by a compiler. Relevant experimental results reveal that the Caliburn processor increases speedup by a factor of 3.08, and reaches a working frequency of 425 MHz using the TSMC 40nm technology library.

Acknowledgments. This work is supported in part by the National Science Council of Republic of China, Taiwan under Grant NSC 101-2221-E-033 -049.

REFERENCES

[1] de Souza A.F., and Rounce, P., Dynamically Scheduling VLIW Instructions. Journal of Parallel and Distributed Computing 60(2000) No. 12, 1480-1511.
[2] Bluespec, Inc., Bluespec SystemVerilog User Guide, 2008. Available on: www.bluespec.com
[3] Ernst D., Hamel A., Austin T., Cyclone: A Broadcast-Free Dynamic Instruction Scheduler with Selective Replay. In: 30th Annual Int. Symposium on Computer Architecture, 2003.
[4] MIPS Technologies, Inc. MIPS32™ Architecture for Programmers Volume I-III: Introduction to the MIPS32™ Architecture, Revision 2.0, 2003.
[5] Conte T., and Sathaye S., Dynamic Rescheduling: A Technique for Object Code Compatibility in VLIW Architectures, In: 28th Annual International Symposium on Microarchitecture, 1995.
[6] Sharangpani H., and Arora K., Itanium Processor Microarchitecture, IEEE Micro, 20(2000), No. 5, 24-43.

***Authors**: Prof. Slo-Li Chu*, Mr. Geng-Siao Li, and Mr. Ren-Quan Liu are with Department of Information and Computer Engineering, Chung Yuan Christian University, 200, Chung Pei Rd., Chung Li, 32023, Taiwan.*
**Corresponding author: E-Mail: slchu@cycu.edu.tw.*