Applied Mathematics & Information Sciences
An International Journal

DynaPack: A Dynamic Scheduling Hardware Mechanism for a VLIW Processor

Slo-Li Chu[*]*, Geng-Siao Li, and Ren-Quan Liu*

Department of Information and Computer Engineering, Chung Yuan Christian University, Chung Li, 32023, Taiwan

Received: Jul 8, 2011; Revised Oct. 4, 2011; Accepted Oct. 6, 2011

Abstract: Continuously growing of semiconductor technology makes the processor architectures more complicated to improve the instruction level parallelism. In the mechanisms of improving instruction level parallelism, VLIW is an attractive technique to improve parallelism without complicated instruction reordering mechanism for dynamic execution. The instruction scheduling is relied on VLIW compiler to select and pack suitable instructions into a VLIW bundle. The programs have to be recompiled accordingly. This drawback limits the popularity of VLIW processors. In this paper, a novel VLIW processor, Avatar, is proposed to overcome the above incompatible problems. By integration with a novel scheduling/packing mechanism, DynaPack, this processor can directly execute the legacy MIPS32 binary codes without recompilation, and fulfill the instruction level parallelism of Avatar VLIW processor. By integrating a new instruction scheduling/packing hardware mechanism, DynaPack can analyze the dependence relations of instructions, maintain their correctness, and pack suitable instructions into a VLIW bundle on-the-fly. The experimental result reveals that Avatar processor with DynaPack mechanism can obtain up to 3.4 instructions per cycle. The chip fabrication results of DynaPack mechanism can achieve 111 MHz by consuming 19343886μm^2 under TSMC 0.13μm technology library.

Keywords: Dynamic instruction packer, VLIW, MIPS32, Bluespec SystemVerilog.

1. Introduction

Continuously growing of semiconductor technology makes the processor architectures more complicated to improve the instruction level parallelism. In these mechanisms for improving instruction level parallelism, VLIW [8] is an attractive technique to improve parallelism without complicated instruction reordering mechanism for dynamic execution, therefore it widely adopted in mathematical and scientific computations. The major characteristic of VLIW architectures is to pack independent instructions into a wider instruction, aka instruction bundle. Since several parallelizable instructions can be executed by the multiple functional units simultaneously, the instruction level parallelism can be improved. Different from conventional dynamic superscalar that adopts complicated mechanisms, such as Tomasulo algorithm, reordering buffer, register renaming [7], VLIW architectures rely on sophisticated compilers to analyze and pack instructions into VLIW bundles in compile-time. It can reduce the hardware cost of above hardware scheduling mechanisms but induces another serious problem. Since the VLIW bundle and conventional instructions are dramatically different, the programs have to be recompiled. This incompatible problem limits the popularity of VLIW architectures. Some researcher proposed dynamically trace scheduled VLIW architecture [1] to solve this problem. It integrates a VLIW engine, a conventional superscalar core into a processor, with a complicated hardware scheduler to keep instruction trace and dispatch the suitable instruction bundle/instructions to the VLIW engine superscalar core, according to the profiling results of scheduler. Due to its complex tracing/scheduling mechanism, this hardware mechanism requires a huge amount of hardware resources and executing time. Also, it can not execute the legacy instructions. The problem of binary incompatibility still reduces its usage.

In this paper, a novel VLIW processor, Avatar, is proposed to overcome the above incompatible problems. By integration with a novel scheduling/packing mechanism, DynaPack, this processor can directly execute the legacy MIPS32 [4] binary codes without recompilation, and fulfill the instruction level parallelism of Avatar VLIW processor. By integrating a new instruction scheduling/packing hardware mechanism, DynaPack can analyze the depen-

[*] Corresponding author: e-mail: slchu@cycu.edu.tw

dence relations of instructions, maintain their correctness, and pack suitable instructions into a VLIW bundle on-the-fly. The detailed mechanism of DynaPack will be discussed in the following sections.

The rest of this paper is organized as follows. Section 2 briefly discusses related works of dynamic scheduling mechanisms for VLIW architectures. Section 3 presents the detailed architecture and execution flow of proposed DynaPack architecture and corresponding scheduling/packing mechanisms. Section 4 shows the experimental results of DynaPack mechanism. Finally, the concluding remark is proposed in Section 5.

2. Related Works

Dynamic scheduling mechanism is a possible solution to overcome binary code compatibility problems in VLIW architectures. Cyclone scheduler [3] is a kind of dynamic instruction scheduler by using time estimation method. If the real execution time of the program meets the estimated time, the scheduler will issue those instructions into functional units. If not, the scheduler will insert instructions to replay queue. This scheduler is suitable for superscalar architectures with out-of-order execution capabilities, because the replay penalty of superscalar with out-of-order execution is less than VLIW architectures.

Dynamically Trace Scheduled VLIW [1] is another kind of dynamic scheduling VLIW architectures. As mentioned before, this VLIW architecture integrates trace scheduler, VLIW engine and a superscalar. The trace scheduler reschedules original instructions and packs into VLIW instructions, and then saves VLIW instructions to VLIW cache. Some original instructions that can't be executed concurrently will save to primary instruction cache. Because of these researches are simulator-based experiments, some hardware latency penalty and extra logic cost is not mentioned. The feasibility of hardware implementation is also hard to distinguish.

The other approach to overcome binary code compatibility problem is Dynamic rescheduling [5][6]. This mechanism translates the source code into new object code to fulfill the requirement of binary compatible. Because dynamic rescheduling mechanism is integrates with operating system and memory management, the scheduling and translating latency may be partially hidden by page fault, but still delay the executing time.

3. The Architecture of Avatar Processor with DynaPack Mechanism

The proposed VLIW processor, Avatar, is quad-issue VLIW architecture with a sophisticated instruction scheduler/packer, DynaPack, to execute the legacy MIPS32 binary codes on-the-fly, without recompilation. The basic pipelined datapath of Avatar processor is as shown in Fig. 1. The scheduling/packing mechanism of DynaPack is discussed later.

The execution flow of Avatar processor is as below. All of the legacy MIPS32 instruction stream are fetched from instruction cache, and then stored into Instruction Buffer for further scheduling and packing. The size of Instruction Buffer in Avatar processor is limited to 16, due to the basic block size of general applications are rarely exceeded 16 instructions. Then DynaPack will schedule the instructions in Instruction Buffer by analyzing the dependence relations of all 16 instructions, and then pack four instructions into a VLIW bundle, for executing on Avatar processor. Since the scheduling and packing mechanism can be reduced as simple hardware logic operations, the complexity of DynaPack is less then the dynamic instruction scheduler, such as Tomasulo algorithm, reorder buffer, in conventional out-of-order superscalar processors. The further execution of packed VLIW bundle can be issued by conventional VLIW pipelined datapath, as showing in Fig. 1. Therefore, the major design objectives of Avatar processor and DynaPack mechanism are (1) improve instruction level parallelism by using simple VLIW mechanism to reduce hardware cost, (2) execute legacy MIPS32 programs by using simple hardware scheduling/packing mechanism without recompilation.

3.1. The Concept of DynaPack Mechanism

In order to execute the legacy MIPS32 binary instructions without recompilation, a hardware instruction scheduling and packing mechanism, DynaPack, is proposed to fulfill the above requirements. DynaPack is able to select four candidate instructions from Instruction Buffer and then pack them into a VLIW instruction bundle in a single cycle. This mechanism is consisted of two major components, Instruction Scheduler and Instruction Packer. The organization of DynaPack mechanism is as shown in Fig. 2.

Instruction Scheduler is composed by Instruction Dependence Analyzer and Instruction Dependence Checker, to analyze the data dependence relations and choose the suitable instructions, respectively. The Instruction Dependence Analyzer determines all kinds of dependence relations of the instructions in the Instruction Buffer and stores the results of the dependence relations into the Instruction Dependence Table. The Instruction Dependence Analyzer requires the detailed attributes of instructions, such as instruction types, instruction formats, operand types, and operand numbers, to identify all dependence relations of the instructions in the Instruction Buffer. The following stages can pack the suitable instructions in to a bundle accordingly.

Before the instructions are stored into Instruction Buffer, Basic Block Detector detects branch/jump instructions firstly. If there is any branch/jump instruction, the following instructions will be marked as invalid and stop fetching instructions. Therefore the Instruction Buffer can keep up to 16 instructions or instructions within a basic block that are bounded by two branch instructions.

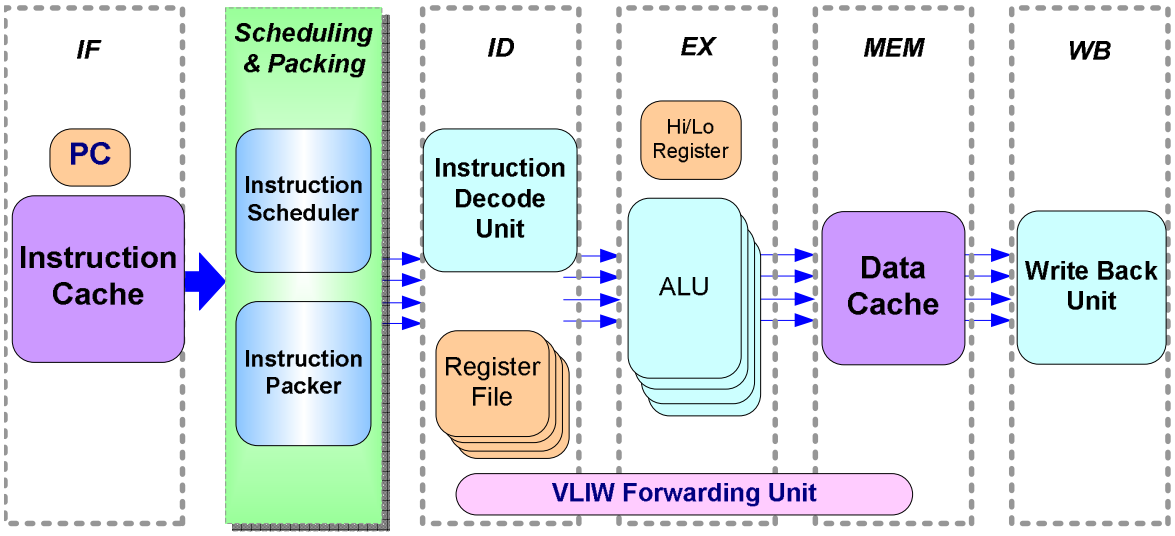

Figure 1 The basic pipelined stages of Avatar processor.

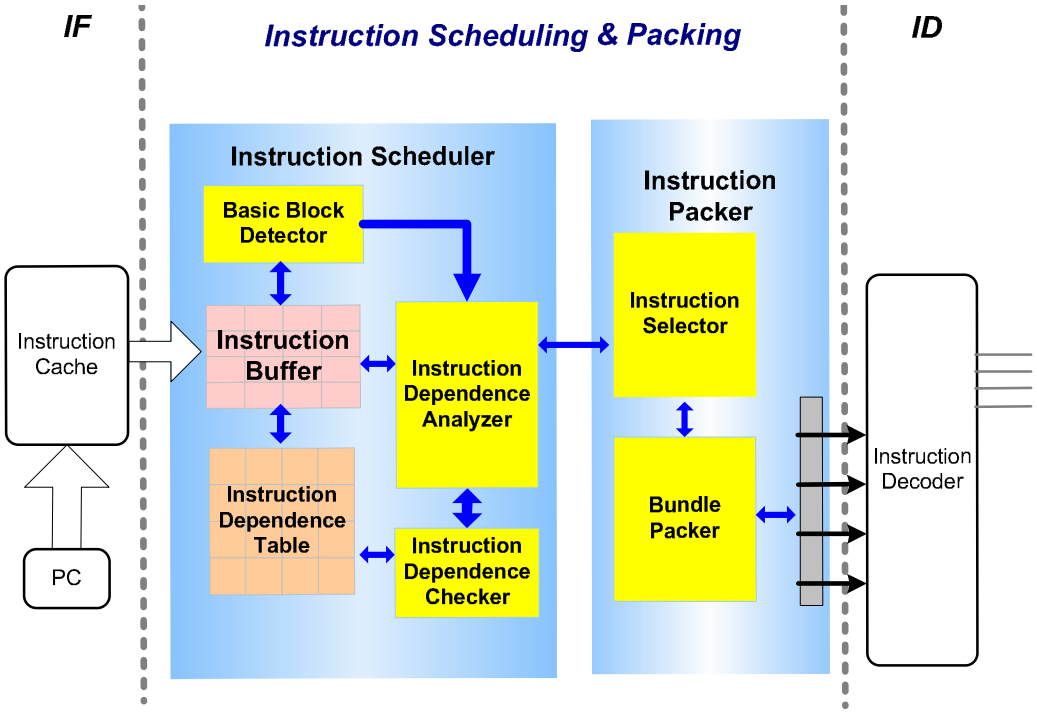

Figure 2 The organization of DynaPack mechanism.

Instruction Packer is composed by Instruction Selector and Bundle Packer for selecting suitable instructions and packing instructions into an instruction bundle, respectively. Each instruction bundle is composed by four MIPS32 instructions that can execute simultaneously. The Instruction Selector firstly analyzes the instructions and picks up the independent instructions from Instruction Scheduler in-ordered. When four instructions are selected, these instructions are sent into Bundle Packer to pack into an instruction bundle. Then the packed instruction bundle can be executed by following pipelined stages of the VLIW MIPS32 processor.

3.2. The Scheduling Flow of DynaPack Mechanism

The analyzing flow of DynaPack mechanism is as shown in Fig. 3. The legacy, unpacked MIPS32 instructions are fetched from instruction cache, and then Basic Block Detector is activated to determine the boundary of a basic block by using two branch instructions. Then the instructions are ignored and stop to fetch consecutive instructions from Instruction Cache to avoid the possible control hazards. Then the screened instructions are stored into Instruction Buffer for the further dependence analysis by Instruction Dependence Analyzer. All the dependence relations of the instructions are figured out, summarized, and stored into Instruction Dependency Table within a single cycle. Then Instruction Dependence Checker is applied to determine all of the candidate instructions according to the results in Instruction Dependency Table, to deal with the data hazards of the instructions. The Instruction Packer can select suitable instructions according to the results in Instruction Dependency Table, and pack the suitable instructions into a single VLIW bundle after considering the possibility of structure hazards. The detailed mechanisms of above scheduling steps are mentioned below.

3.3. The Structure of Instruction Buffer in DynaPack Mechanism

The organization of Instruction Buffer is as shown in Fig. 4. Instruction Buffer can store up to 16 conventional MIPS32 instructions in a single cycle, so it contains 16 entries, which consists of 8 fields to the information of an instruction. In addition to the required fields of the instruction, it also includes a Valid Bit to represent the current status of the corresponding instruction for the following analysis. The instruction marked as ”Invalid” will be ignored at this time when scheduling and packing instructions. The instructions stored in Instruction Buffer are in-order and scheduled by first-in-first-out policy. When the Instruction Buffer is empty, the instructions of next basic block are processed and begin next round of instruction scheduling.

3.4. The Functionality of Basic Block Detector in DynaPack Mechanism

Since the branch instructions can change the execution flow of the program, the fetched and analyzed instructions in Instruction Buffer will become invalid. Therefore a prescreened mechanism, Basic Block Detector, is required to identify the boundary made by two branch instructions (aka Basic Block), and reduces the redundant analysis of invalid instructions.

Due to the characteristic of delay slot in MIPS32 instruction set, except Branch-likely instructions, the consecutive instruction of the branch instructions will be executed. The executed result of the instruction in the delay slot is independent with the decision of the corresponding branch instruction. However, the delay slot instruction of the Branch-Likely instructions will be executed it the branch is taken. Therefore the proposed Basic Block Detector has to be identified carefully.

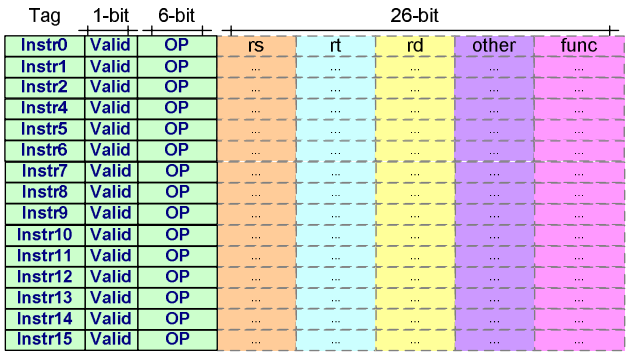

Figure 4 The structure of Instruction Buffer in DynaPack mechanism.

According to the above description, the main functionality of Basic Block Detector is to determine the branch instructions when fetch legacy MIPS32 binary instructions into Instruction Buffer. If a branch instruction is found in the fetched instruction stream, the consecutive instructions, include the instruction in the delay slot, are marked as ”Invalid”, and the process of filling Instruction Buffer will be terminated. Then Basic Block Detector will determine the correct instruction for the next round of filling Instruction Buffer, according to the result of branch taken and the branch target address (aka updated PC address). The instruction in the branch delay slot will be arranged to the first position of Instruction Buffer in the next round. Fig. 5 illustrates the determination flow of Basic Block Detector.

3.5. The Analyzing Flow of Data Dependence by Instruction Scheduler

When unpacked instructions filled into Instruction Buffer, Instruction Scheduler begins the processes of analyzing, scheduling, and packing. Firstly, Instruction Dependence Analyzer is activated to analyze the dependence relations of the instructions in the Instruction Buffer. There are three kinds of possible data dependence relations: true dependence, anti dependence, and output dependence. The analyzing mechanisms of these three dependence relations are described below.

True data dependence, aka RAW (Read after Write) data hazard, occurs when the following instruction(s) require data which is updated by the current instruction. When Instruction Packer packs instructions into a VLIW bundle, the instructions can not be packed in the same bundle if

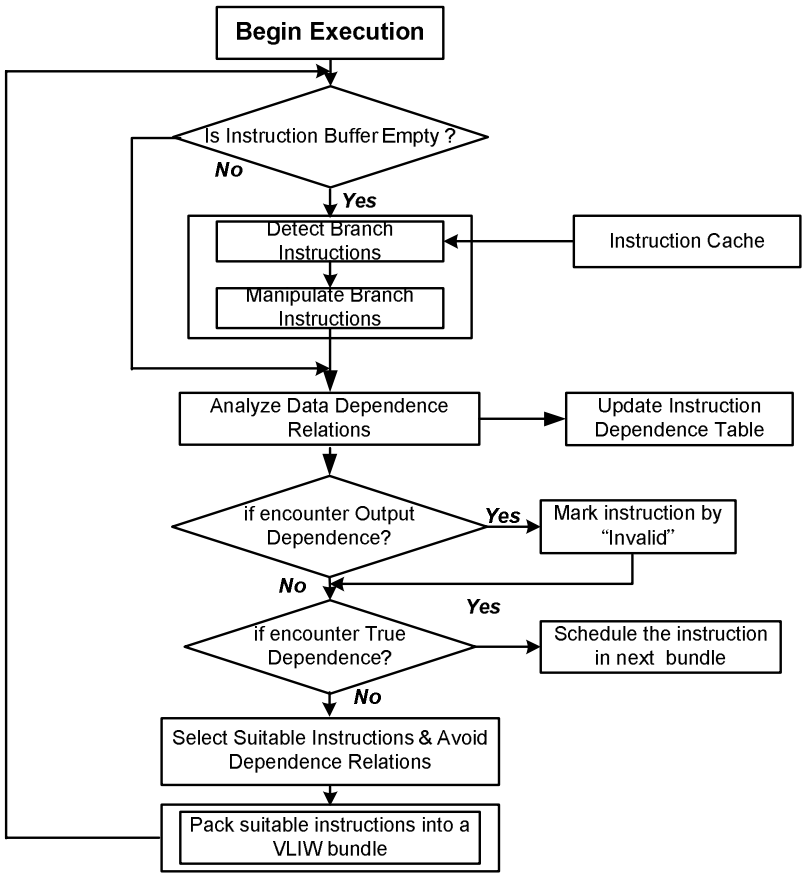

Figure 3 The analyzing/scheduling flow of DynaPack mechanism.

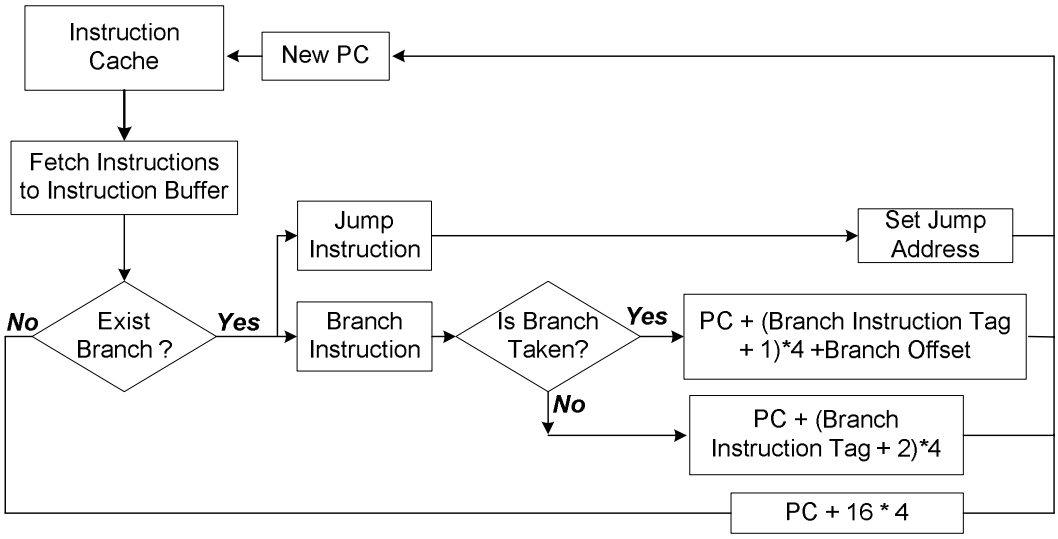

Figure 5 The manipulation steps of branch instruction by Basic Block Detector.

there is any true dependence between them. These dependent instructions must be arranged into different bundles by their original lexicographical orders. The pending cycles of these dependent instructions can be reduced by proposed VLIW Forwarding Unit.

Anti data dependence, aka WAR (Write after Read) data hazard, occurs when the following instruction(s) updated data which has been read by the previous instruction. Since the required data has been updated in the precious bundles, packing anti dependent instructions into the same bundle will not affect the correctness.

Output data dependence, aka WAW (Write after Write) data hazard, occurs when the following instruction(s) and current instruction update the same data. According to the rule of "last value assignment", the last dependent instruction will affect the computing result. All of the previously dependent instructions can be omitted. It can reduce the total code size by removing these redundant instructions in this situation.

Instruction Dependence Analyzer is basically based on the comparison the register number fields of the instructions in Instruction Buffer. The comparison operations can be reduced as a simple XNOR with reduced-and logical operations. Therefore the analyzing operations of all 16 instructions in Instruction Buffer can be completed in a cycle. The analyzing results are stored into Instruction Dependence Table for further analyzing stages.

3.6. The Organization of Instruction Dependence Table

The analyzing results dependence relations by Instruction Dependence Analyzer are stored into Instruction Dependence Table. The organization of Instruction Dependence Table is shown in Fig. 6. The number of the entry in the table denotes the lexicographical order of the instruction, and the less number denotes the higher execution order. The cell cross by Instruction i (Instr i) and Instruction j (Inst j) represents the dependence relations from Instruction i and the following Instruction j. Each cell consists of three bits data to represent true dependence, anti dependence, and output dependence, respectively. Therefore the dependence determination can be reduced as simple logic operations and can be completed within a clock cycle.

3.7. The Organization of Instruction Dependence Table

The scheduling algorithms of DynaPack mechanism can be divided into two parts, instruction scheduling algorithm consists of two parts, Output Dependence Filter, and True Dependence Detector, as listed in Fig. 7. They can determine the output dependence and true dependence of each instruction and mark the corresponding flags of the instructions in Instruction Dependence Table.

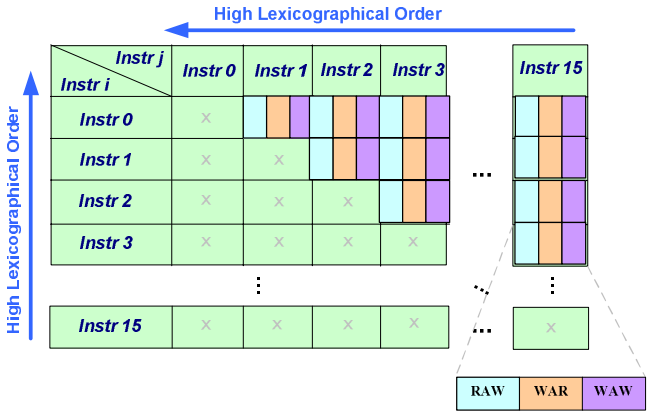

Figure 6 The structure of Instruction Dependence Table.

Algorithm 1: Output Dependence Filter

i, j: instruction number, i<j 0<=i,j<16
C_i : count number of output dependence instruction, size 16

if(ID_{ij}[2] equal 0)
  $C_i = C_i + ID_{ij}$[0];

if(C_i>0)
  mark instruction i "Invalid"

Algorithm 2: True dependence Detector

i, j: instruction number, 0<=i,j<16
T_j : value of executed reduce-or operation from 0 to i in every column j, size 16.

T_j = reduce-or (ID_{ij});

if(T_j>3)
  schedule instruction J in next bundle
else
  send instruction j to Bundle Packer

Figure 7 The algorithms of scheduling instructions in DynaPack mechanism.

The functionality of algorithm 1 is to scan all the instructions in Instruction Buffer, and mark the corresponding field as "invalid" if the following instructions in the Instruction Buffer have output dependence with instruction i. After accumulating the counts of output dependences from the following instructions, Ci will be stored into the corresponding field of instruction i in Instruction Buffer. All instructions will be scanned and checked. If Ci is greater than zero, it means some instruction have output dependence with instruction i, Output Dependence Filter algorithm will mark instruction i as "invalid".

The objective of algorithm 2 is to detect the true dependence and select the suitable instructions into Bundle Packer stage. If an instruction j has true dependence with other instructions, the exactly instructions that have true dependence with the instruction j can be ignored, it only needs to count the frequency of true dependence, as listed in algorithm 2. It adopts simple reduce-or operation to calculate the relationships between instruction j and instruction 0 to j-1, and use Tj saved the result. If Tj is great than three (Tj's 2nd bit is 1), it means some previous instructions have true dependence with instruction j. The instruction j would be scheduled in next bundle, and omitted in current bundle.

4. Experimental Results

According to the above discussion, the Avatar VLIW processor and DynaPack scheduler is composed by sequencer units, scheduler units, detector units, buffering units, and a lot of decision rules. These hardware modules require several kinds of handshaking protocols, and make the design and debug more difficult. According, a new data-oriented design methodology is constructed to overcome these questions. This methodology focuses on manipulating data and the dependence of source operands that generate the results, instead of controlling precise timing and signals. Conventional hardware datapath that consist of control signals, multiplexor, and dedicated functional units can be replace by the data manipulating mechanism, called "rule", and the simple handshaking mechanism, called "interface method". Since the main consideration is the states of data, instead of datapath timing, the difficulties of complex chip design can be reduced by approaching the nature of algorithm and shorten the design cycle. Conventional hardware description languages, such as Verilog, and VHDL, are not suitable for designing hardware by proposed data-oriented methodology. Therefore, the proposed quad-issued VLIW processor, Avatar, and accompanied a sophisticated hardware instruction scheduler/packer, DynaPack is developed by a new hardware design language, Bluespec SystemVerilog [2].

Bluespec SystemVerilog (BSV), developed by MIT, is based on a synthesizable subset of SystemVerilog. The basic building block of BSV is rule. Instead of synchronous always blocks, rule can achieve correct concurrency and eliminating race condition. Each rule can be viewed as a declarative assertion expressing a potential atomic state transition. The BSV compiler produces efficient synthesizable RTL Verilog codes that manage all the potential interactions between rules by inserting appropriate arbitration and scheduling logic, called handshaking circuits. The atomicity of rules can avoid unwanted race condition in large designs. Therefore BSV is suitable for designing complex algorithms by above data-oriented methodology and can generate synthesizable Verilog design quickly.

After designing Avatar VLIW processor with DynaPack mechanism by using BSV, the improvement of instruction level parallelism can be evaluated. Fig. 8 illustrates the relation between varied sizes of loop body in the program and effects of instructions per cycle (IPC), which is an important metric of instruction level parallelism to denote the concurrent instructions that Avatar VLIW processor can complete. According to the experimental results, if the size of loop body is more than 4, the IPC can larger than 1. The advantage of Avatar VLIW processor can be obtained. Reminding that the input benchmark is not compiled by VLIW compiler, the IPC improvement is improved by DynaPack mechanism which analyzes and packs conventional MIPS32 binary instructions on-the-fly. The results also reveal that DynaPack can achieve overall IPC up to 3.4 if the size of loop body is large enough. It also demonstrates the capabilities of proposed DynaPack mechanism that can improve the IPC of the conventional VLIW processor; even the software programs haven't been recompiled.

The Avatar VLIW processor with DynaPack mechanism is designed by Bluespec SystemVerilog, and then generated high quality synthesizable Verilog codes. The functional correctness of Verilog version Avatar and DynaPack has been verified by using Synopsys VCS Verilog simulator. After synthesizing by Synopsys Design Compiler with TSMC 0.13μm technology library, the fabrication results show that DynaPack mechanism can obtain the working frequency of 111 MHz and consume 19343886μm^2 area size. This fabrication results also show that DynaPack mechanism can be integrated with any modern VLIW processor to improve the IPC and binary compatibility, and will not become the performance bottleneck or charge too much chip area.

5. Conclusions

This paper proposes a novel VLIW processor, Avatar, which can overcome the incompatible problems of conventional VLIW architectures. By integration with a novel scheduling/packing mechanism, DynaPack, this processor can directly execute the legacy MIPS32 binary codes without recompilation, and fulfill the instruction level parallelism of Avatar VLIW processor. By integrating a new instruction scheduling/packing hardware mechanism, DynaPack can analyze the dependence relations of instructions, maintain their correctness, and pack suitable instructions into a VLIW bundle on-the-fly. The detailed mechanism of DynaPack is discussed. According to the experimental result, Avatar processor with DynaPack mechanism can obtain up to 3.4 instructions per cycle. The chip fabrication results of DynaPack mechanism can achieve 111 MHz by consuming 19343886μm^2 under TSMC 0.13μm technology library.

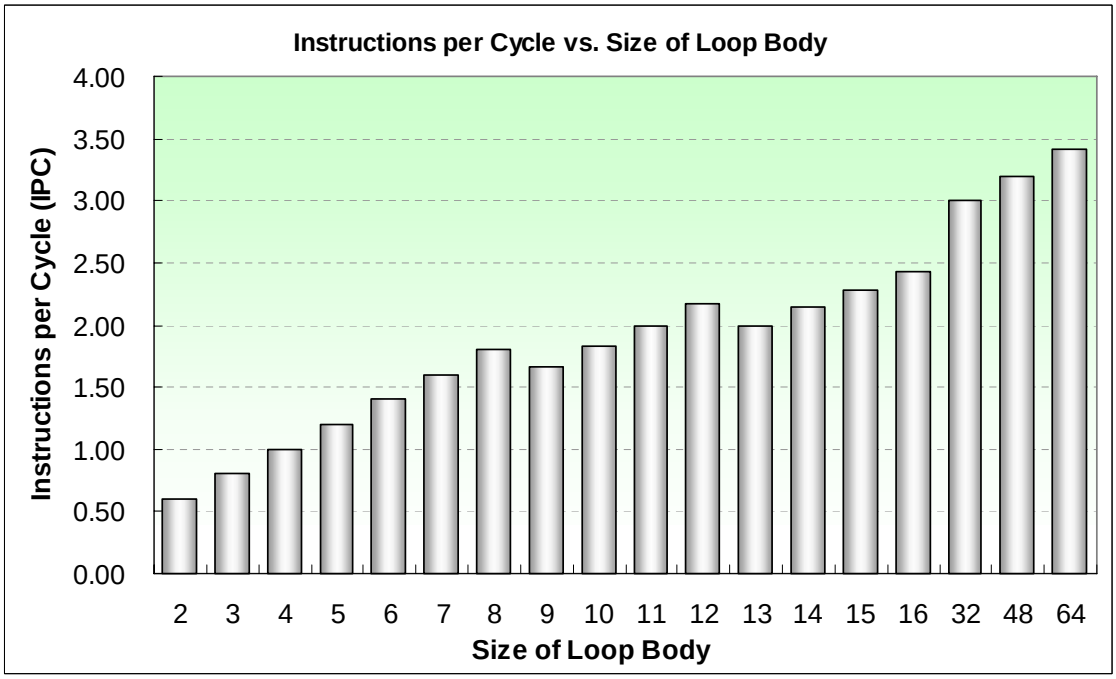

Figure 8 The results of instructions per cycle vs. different size of loop body.

Acknowledgement

This work is supported in part by the National Science Council of Republic of China, Taiwan under Grant NSC 100-2221-E-033-043.

References

[1] A. F. de Souza and P. Rounce, Journal of Parallel and Distributed Computing, 60, 1480 (2000)
[2] Bluespec, Inc. Bluespec SystemVerilog User Guide, (2008). www.bluespec.com.
[3] D. Ernst, A. Hamel, and T. Austin, Proc. 30th Annual International Symposium on Computer Architecture, 253 (2003)
[4] MIPS Technologies, MIPS32 Architecture for Programmers Volume I-III: Introduction to the MIPS32 Architecture, Revision 2.0. (2003). www.mips.com.
[5] T. M. Conte, and S. W. Sathaye, Proc. 28th Annual International Symposium on Microarchitecture. 208 (1995)
[6] M. Jayapala, F. Barat, T. Vander Aa, F. Catthoor, H. Corporaal, and G. Deconinck, IEEE Trans. Computers, 59, 672 (2005)
[7] M. Gupta, F. Sanchez, and J. Llosa, IEEE Trans. Computers, 59, 385 (2010)
[8] J. A. Fisher, IEEE Trans. Computers, 17, 45 (1984)

Slo-Li Chu received his PhD degree in Electrical Engineering from National Sun Yat-sen University in 2002. From August 1998 to 1999, he visited Department of Computer Science, University of Illinois at Urbana-Champaign for one year. He is currently an assistant professor of Department of Information and Computer Engineering, Chung Yuan Christian University, Taiwan. His research interests include computer architectures, parallelizing compilers, system level modeling, system-on-chip design, and GPU architectures.

Geng-Siao Li received his BS degree in Information and Computer Engineering from Chung Yuan Christian University, Taiwan, in 2006. He is currently pursuing for his PhD degree in Electronic Engineering at Chung Yuan Christian University, Taiwan. His research interests include computer architectures, system level modeling, and system-on-chip design.

Ren-Quan Liu received his BS degree in Information and Computer Engineering from Chung Yuan Christian University, Taiwan, in 2006, and the MS degree in Information and Computer Engineering from Chung Yuan Christian University, Taiwan, in 2010. His research interests include computer architectures, system level modeling, and system-on-chip design.