C-030

Superscalar Microprocessor Design in a Hardware Software Co-learning System

HOANG ANH TUAN †, KOICHIRO NAKAMURA‡, KATSUHIRO YAMAZAKI †, AND SHIGERU OYANAGI‡

Graduate School of Science and Engineering, Ritsumeikan University.

1. Introduction

The hardware software co-learning system helps the users to understand the structural design of processors as well as the interaction between processor architecture and software development. To achieve these objectives, the system should contain Compiler, Variable processor simulator and several designs of the processors [1] [2]. This paper describes the design and implementation of a superscalar processor for the co-learning system, and explains the instruction policy, the technique, and the mechanism that is used to design this superscalar processor.

2. Hardware software co-learning system

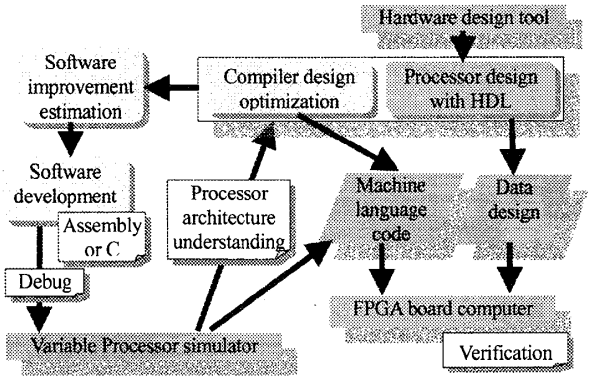

Figure1: Hardware software co-learning system overview

Figure 1 shows the Co-learning system. It can help the users easily to understand the interactions between the hardware and software design in a computing system, be able to work with SoC or embedded projects. The four microprocessor architectures (single cycle, multi-cycle, pipeline, and superscalar architectures) will be implemented in the system, located in the Variable Processor Simulator and FPGAs board computer.

The instructions used here is the MONI instruction set [2] with 42 instructions, and support for the three operand (two sources and one destination) operations. The instruction is 16 bits length with the first five bits for OPCODE. The operand is identified by three bits each, be able to name one among eight internal registers. The set is divided in to four types of instructions: R-type, I5-type, I8-type and J-type [2]. It has three addressing modes: immediate, register, and register indirect addressing modes

3. Design of the superscalar microprocessor

3.1 Datapath

Figure 2 shows diagram design of the processor with the datapath for fetch, pre-decode, decode and execute the instructions, two instructions at a clock cycle. The design contains Arbitrator, decoder, ALU, register file, SP controller, PC controller, and instruction cache. Core of the microprocessor is single cycle architecture processor with decoder and ALU. To crack the computational resources conflict, increasing the machine parallelism, the resources duplication technique has been used with two functional units, unit A and B.

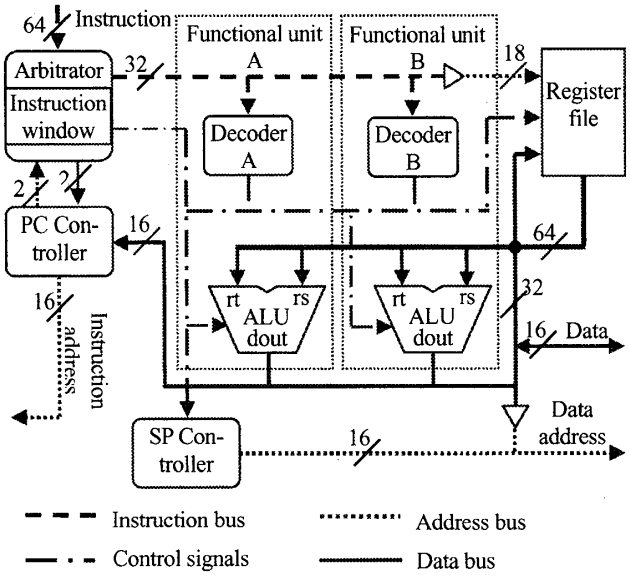

Figure 2: Block diagram of Superscalar microprocessor

The instructions are fetched into the arbitrator at the positive edge of the clock while the Data memory access happens at the negative edge. After that, the register write back operation will occur at the following positive edge of the clock. By that, an instruction can be executed in the time of a clock cycle, forms the single cycle architecture for the core of the microprocessor. In the two functional units, unit A has higher priority than unit B. Supporting for the computing ability, the data bus is designed to transfer four data (64 bits) out and two data (32 bits) in of register file. Instruction cache has transparently included by widening the instruction bus between Instruction memory and Arbitrator to four instructions (64 bits), increasing the ability of fetching up to four instructions simultaneously.

Operations and datapart of the microprocessor is as follows. The instruction memory is active and releases four instructions simultaneously to the Arbitrator for pre-decode at the positive edge of the clock. The Arbitrator depends on its instruction window, takes two instructions among the four received, analyses them in order to find the data dependency as well as the resources conflict if available, and decide if the low priority instruction can be executed. The result then used to generates two instructions and control signal to the functional units A, B and PC controller. At the two functional units, the instructions are decoded and the control signals are sent to the ALUs, register file, data memory, SP controller to control the computing process as well as the data write back action. If the data dependency occurs, the output data from unit A will be forwarded to its destination register and be forwarded to the input of unit B. At the negative edge of the clock, data will be written to the memory (if necessary). The register file is active at the next positive edge of the clock for data write.

3.2 Arbitrator

The role of the arbitrator is executing condition analysis. It decides the instruction issue and the parallelism of processor. In this design, the Arbitrator checks the data dependency, resources con-

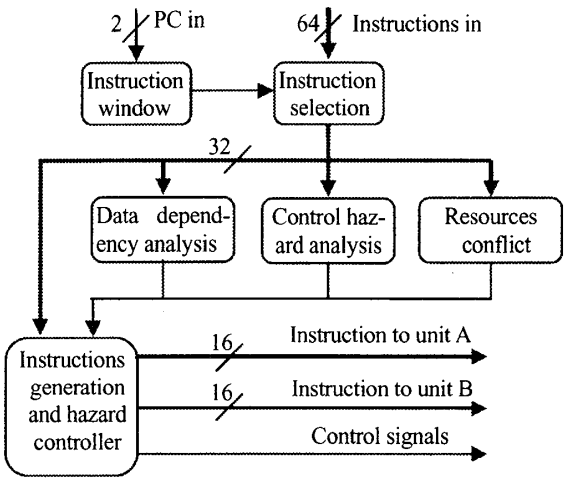

Figure 3: Design of the Arbitrator

flict, and generates instructions as well as control signals for the two functional units. As shown in Figure 3, it has an instruction window, controlled by two least significant bits from the instruction address, used to identify the two necessary instructions among the input of fours. Those instructions will be analyzed to recognize if the data dependency, resources conflict or control hazard occur. Then, control signals will be given to PC controller, two acceptable instructions and some control signals too will be given to the functional units. If any conflict occurs, unit B will be dismissed (by taking the no operation instruction, generated by the Arbitrator). Rules that apply to analyze data dependency, resources conflict are followed.

- Control hazard is determined by comparing the opcode of instruction in unit A with opcode of jump and call operations. If the condition meets, the instruction in the next address cannot be executed. Unit B will receive and execute the no operation instruction.

- Destination conflict is met if the comparison between the destination registers of functional units A and B gives the positive result. If it occurs, data that generated from unit B will over write the data generated by unit A, achieved by disable the write enable signal from the decoder of unit A.

- Data dependency analysis is performed by comparing the destination of unit A with the sources of unit B. If they are equal, data dependency occurs and be solved by data forwarding technique, in which, output of unit A will immediately forward to input of unit B . Results generated by the two units then be written at the same time.

- Storage conflict happens if the opcode of the two instructions are load or store opecode. The external data bus, with its limitation of 16 bits width, allows only one word transfer through. The higher priority instruction in unit A will take advantage and completes in that cycle, while the unit B executes the nop operation.

- Instruction cache miss is the situation that the Arbitrator cannot find the corresponding instruction to execute in functional unit B, occurs when instruction window get the binary 11 value. It occurs as a bad effect of the virtual instruction cache policy. If it happens, unit A takes the instruction identified by the instruction window, while the nop instruction will be generated by the Arbitrator for the functional unit B.

3.3 Other units

- Decoder decodes the instructions given by the Arbitrator, and gives the control signals to other parts of the processor to make them work correctly as the requirement of the instruction.
- ALU receives immediate data given from the instruction or Register file, and control signals from the decoder, generates the output data with some arithmetic or logic operations. The output then sent to the data bus for writing to its destination.
- Register file contains 8 registers, 16 bits each. The location of one register can be identified through three bits address. It can simultaneous give out four data (64 bits), two for one functional unit, and write back two data (32 bits) from the internal data bus depends on the input addresses and the control signals from the decoders.
- Stack Pointer (SP) controller is used to support for the sub-routine operations (push, pop, call and return instructions). SP shows the top of the stack.
- Program Counter (PC) controller is used to control the fetching sequence of instructions from the Instruction memory to execute in the processor.

4. Implementation and simulation result

The design has been written in Verilog HDL with behavior description model, then synthesized and simulated using Foundation ISE and ModelSim tools, implemented on a RC100 FPGAs board. It is written in one and a half month with around 1000 lines. Table 1 shows the hardware evaluation of the design.

Table 1: Hardware cost evaluation

	Number of gates	Number of Flip Flops	Number of LUTs	Number of TBUFs
Arbitrator	9949	14	221	0
Decoder	4592	0	96	0
Register file	27381	128	618	0
ALU	32143	0	672	0
PC controller	5697	24	122	0
SP controller	3401	24	74	0
All MPU	106292	213	2379	186

The clock cycle reduction in simulation with "Bubble sort", "Selection sort" and "Sum up" programs ranged from 36% to 40% depends on the hazard of each program. The bottle-neck occurs mostly in branching and instruction cache misses.

This design will be used for the learners to study processor design with HDL in the co-learning system. They can use the available code and design partly or fully in their own learning project.

5. Conclusion and future work

The superscalar microprocessor design has contributed to the variable processor simulator and FPGAs board computer in the hardware software co-learning system. The in-order issues with out-of-order completion instruction policy, resources duplication and data forwarding techniques have been introduced through the design; helps the learners to understand how to implement those techniques in a superscalar processor. The design should be simulated with some more programs, and be upgraded with cache to increase the cache hit rate. Moreover, the Simulator for the superscalar microprocessor will be written.

References

[1] Mutsumi Oyagi et al. "Design of a Variable Processor Simulator in a Hardware/Software Co-Learning System", Proceeding 66th Information Processing Society of Japan Record, 5T-6, 2004.

[2] Nobuhisa Ikeda et al. "Design of a FPGA Board Computer in a Hardware/Software Co-Learning System", Proceeding 66th Information Processing Society of Japan Record, 5T-5, 2004.

[3] Koichiro Nakamura et al. "Development of the FPGA board computer system for processor architecture education", submitted to FIT, 2004.