PQPpfB: Parallel Queue Processor Architecture in Verilog-HDL

ben a. abderazek ,† Markovskij Arsenji ,† Kazuyuki Kiuchi ,†
Musfiquzzaman Md. Akanda ,† soichi shigeta ,†
tsutomu yoshinaga † and masahiro sowa †

In this article, we describe the hardware implementation of a Simple parallel produced order queue processor (PQPpfB) architecture that stores data in produced order scheme. Data is inserted in the queue in produced order scheme and can be reused. This feature has a profound implication in the areas of parallel execution, programs compactness and hardware simplicity. We first give an overview of the processor architecture. Then, we give the implementation method and the preliminary evaluation results.

1. Introduction and Motivations

Nowadays, the shifts in hardware and software technology force designers and users to look at micro architecture that process instructions stream with high performance, low power consumption, and short program length. In order to achieve high performance, micro architecture research has emphasized instruction-level parallelism processing, which has established in superscalar architecture without major changes to software. Since the program contains no explicit information about available ILP, it must be discovered by the hardware, which must then also construct a plan of action for exploiting parallelism. In short, computers haves thus far achieved this goal at the expense of tremendous hardware complexity − a complexity that has grown so large as to challenge the industry ability to deliver ever-higher performance.

The possibility of designing a processor architecture that could offers simple hardware, high performance and small code size, while maintaining other characteristics is what led us to develop a parallel queue processor architecture, which stores data in a FIFO registers (refer to Fig. 2) and exploits parallelism dynamically. Since the Queue word is designated implicitly in the instruction, the instruction length becomes short and it has independence from the physical operand queue word. Moreover, since the assignment of queue word is based on a single assignment rule (SAR), WAR hazard does not occurs.

† Graduate School of Information Systems, The University of Electro-Communications, Tokyo, Japan

In this paper we propose a Simple parallel produced order queue processor (PQPpfB) architecture that stores data in produced order and offers much lower hardware complexity than RISC, STACK or our earlier proposed queue processor architecture[1),2)].

2. PQPpfB System Architecture

The PQPpfB architecture has six pipelining stages as described bellow:
(1) Fetch: 12 bytes are fetched from the memory and inserted into a fetch buffer. The address used to access the memory is also used to access a branch target buffet (BTB). These accesses are done at the same time. The accessing address hits in the BTB, the information stored in the BTB is known right after the access. The branch type (kind) can be used to choose the prediction source before the branch is decoded (refer to Fig. 2); (2) Decode: decode the function and the operands; (3) Queue computation: Calculate the queue head and tail values; (4) Barrier Queue: Insert Barrier; (5) Issue: Find executable instructions and issue them; and (6) Execution: Execute instructions and update queue registers and memory. The queue computation stage computes QH and QT values of each instruction. They are the value when each instruction is to be executed in serial. The issue stage checks the data in the queue that each instruction requests and checks the execution unit availability. Then it issues the instruction if there are. The architecture of the PQPpfB processor is shown in Fig. 1. It consists of a fetch Unit (FU), a decode unit (DU), a Queue computing Unit (QCU), a Barrier Queue Unit (BQU), an issue unit (IU), and an execution unit (EU).

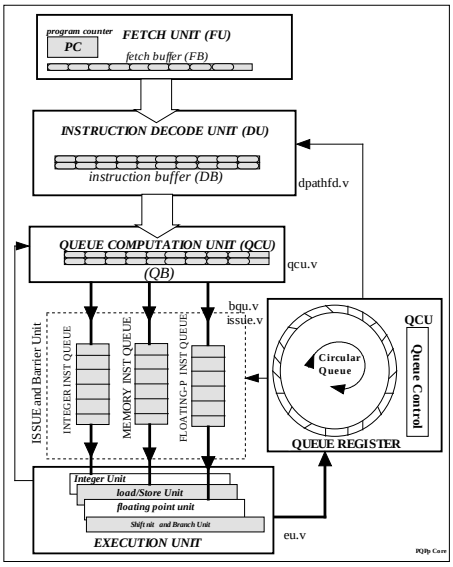

Fig. 1 System Architecture

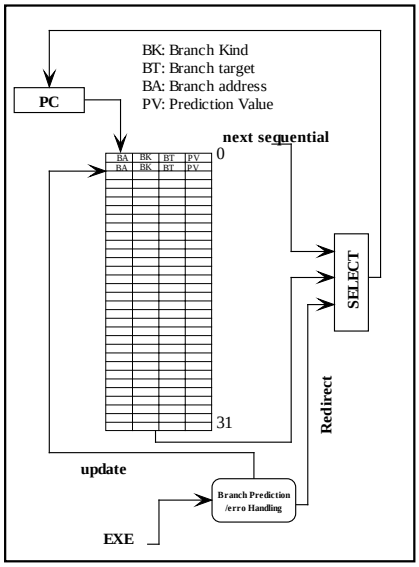

Fig. 2 Branch Instructions Handling Mechanism

3. Implementation Results

The processor was designed in RTL model and successfully integrated and simulated with Synopsis Verilog XL simulator. Figure 4 describes the modules which were designed and integrated to form the processor core.

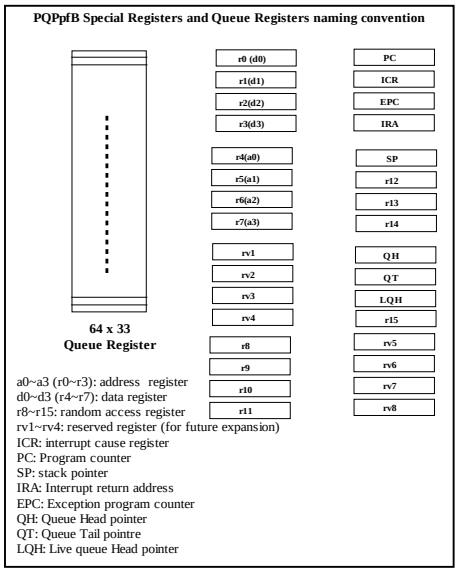

Fig. 3 Hardware Registers

Source	Module Description
inst_define.v	instructions definition (119 instructions)
top.v	top file module
dpathfd.v	data path, fetch and decode module
qcu.v	queue computation module
bqu.v	barrier queue module
issue.v	instruction issue module
eu.v	instruction execution module
Total verilog HDL code: 2275 (ln)	

Fig. 4 Verilog HDL code Implementationn of the PQPpfB processor. The whole code was integrated and the netlists for the main modules successfully exported with Synopsis FPGA Compiler II tool.

4. Conclusion and Future Work

In this article, we presented an overview of a novel simple parallel Queue processor architecture (PQPpfB) that uses queue register for operand and results manipulations. The processor was designed in RTL model and successfully integrated and simulated with Synopsis Verilog XL simulator. Our Future work is to verify its functionality and overall performance in an Altera APEX FPGA board and realistic bencmark programs.

References

1) PQPpfB project: http://www.sowa.is.uec.ac.jp
2) Sowa M., B. A. Abderazek, and al.: Proposal and Design of a Parallel Queue Processor Architecture (PQP), 14th IASTED Int. Conf. on Parallel and Distributed Computing and System, Cambridge, USA,pp.554-560 (2002).