A Novel Instruction Set for the Packet Processing on the Network ASIP

Won-young Chung[1], Yeo-phil Yoon[2], and Yong-surk Lee[3].
School of Electrical and Electronic Engineering, Yonsei University
134 Shinchon-dong, Seodaemun-gu, Seoul 120-749, Korea
Tel: +82-2-2123-2872, Fax: +82-2-312-4584,
E-mail: {[1]wychung, [2]ypyoon}@dubiki.yonsei.ac.kr, [3]yonglee@yonsei.ac.kr

Abstract: In this paper, we propose a new instruction set for a network ASIP(Application Specific Instruction-set Processor). The new instruction set was designed for the packet processing engine on a network router. The network ASIP to accelerate the packet processing operation was built on a baseline ASIP, which is based on the general RISC structure. The new instruction set is divided into two groups. They are operated on each of its functional unit within the execution stage. After the derivation of the HDL-model from LISA, the functional units were replaced by a hand-written Verilog-HDL.

1. Introduction

Recently, research on network have been focused on BcN(Broadband convergence Network), and ALL-IP convergence is the main trend in terms of the traffic character [1][2]. Also, due to the development of IPTV, VoIP, and VOD, multimedia packets have increased within total network traffic. Due to the increase of multimedia packets, the total network traffic, and the amount of payload, the information has to be processed is growing gradually. Because of the number of IP packets and the limitation of the speed of the physical network, network congestion occurs frequently. To improve the performance, we should arrange a processor on each port of the router. By designing the processor for the router, an ASIP has more energy efficient network functions than general processors or ASICs. Moreover, it is easily extended by upgrading new software.

In this paper, we have designed a new instruction set and its hardware blocks to accelerate the packet process. We have named the designed hardware blocks NX(Network eXtension) and the new instruction set uses it.

The contents of this paper are as follows. Section 2 explains the fundamental basics of the baseline ASIP used in this paper and the ASIP with the NX for the network specific ASIP. Also, explanations on the NX ISA (Instruction Set Architecture), its operation, and its functional unit structure are included. Section 3 describes simulation results on the baseline ASIP and the proposed network ASIP. Finally, Section 4 gives the concluding remarks.

2. ASIP Design Flow

2.1. General ASIP

The general ASIP was designed based on a 5-stage pipeline 32-bit MIPS-DLX RISC(Reduced Instruction Set Computer) architecture, as shown in Fig. 1. The EX (execution) stage consists of functional units and the functional units consist of two ALUs, one shifter, and one multipler. It can operate basic arithmetic & Logical instructions, branch instructions, and load & store instructions, and so on [3].

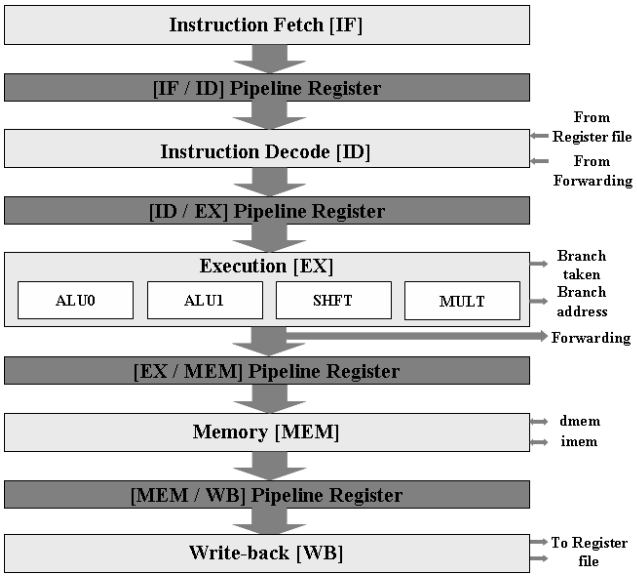

Fig. 1 General ASIP

2.2. Network ASIP

New instructions and the corresponding hardware blocks for packet processing are added to the general ASIP. In Section 2.2.1, the details of the new instructions are explained, and in Section 2.2.2, the hardware blocks of the new instructions are explained.

2.2.1 Proposed instruction sets (NX)

Packet communication based routers have to process many packets within limited time. Comparing the specific bits of the parcket header to the layer 2 MAC table is, generally, the most frequently used operation. To accelerate the operation, we have added new instrucions for bit manipulation (SPECIFIC) and mask compare(COMPARE). Table 1 shows the brief explaniations of the new instructions.

- **SPECIFIC Instruction set group**

The SPECIFIC Instruction set group is comprised of set, clear, test, and flip instrucions. Fig. 2 defines the instruction format of the instruction group. The src_reg 2(the first bit of the specific region) and the src_reg 3(the last bits of the specific region) represent the specific region of the src_reg

1, and the results of its operation is stored in the dest_reg through the MUX. Fig. 3 shows the operation of the test instructions in the SPECFIC instruction group.

- **COMPARE Instruction set group**

The COMPARE instruction set group is comprised of cmp, clz, cmpclz, mcmp, and mcmpclz, which are shown in Fig. 4. The clz instruction has one source register, the cmp and the cmpclz instruction has two source registers, and the mcmp and mcmpclz have four source registers. Fig. 5 shows the operation of mcmpclz which is in the COMPARE instruction group. The mask1 and mask2 is the src_reg3 and src_reg4, respectively, and will be logically AND operated with src_reg1 and src_reg2, respectively. And after the AND operation, the results result1(AND operation on src_reg1 with src_reg3) and result2(AND operation on src_reg2 with src_reg4) will be compared with each other and will return the tailing zero value of the compared result between result1 and result2.

Table 1. NX insturctions

NX Instruction set group	Instruction	Description
SPECIFIC	set	Sets the specific region to '1'. rs2 : starting point, rs3 : ending point
	clear	Clears the specific region to '0'. rs2 : starting point, rs3 : ending point
	test	Bitwise comparasion of specific region, and sets rd=0 when all of the bits are equal, and rd=1 when non-equal. rs2 : starting point, rs3 : ending point
	flip	Bitwise flip operation of specific region. rs2 : starting point, rs3 : ending point
COMPARE	cmp	Bitwise comparasion of rs1 and rs2, and sets rd =0 when all of the bits are equal and rd=1 when non-equal.
	clz	Counts the leading zeros.
	cmpclz	cmp + clz
	mcmp	Mask operation on Rs1 and Rs3, on Rs2 and Rs4. And compare the result of the two masked operation.
	mcmpclz	mcmp + clz

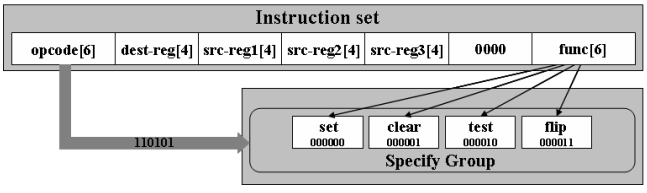

Fig. 2 SPECIFIC Instruction set of N-Extension

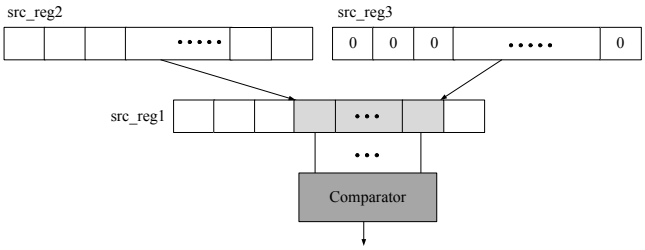

Fig. 3 test instruction

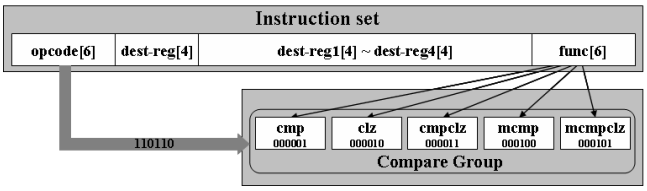

Fig. 4 COMPARE Instruction set of N-Extension

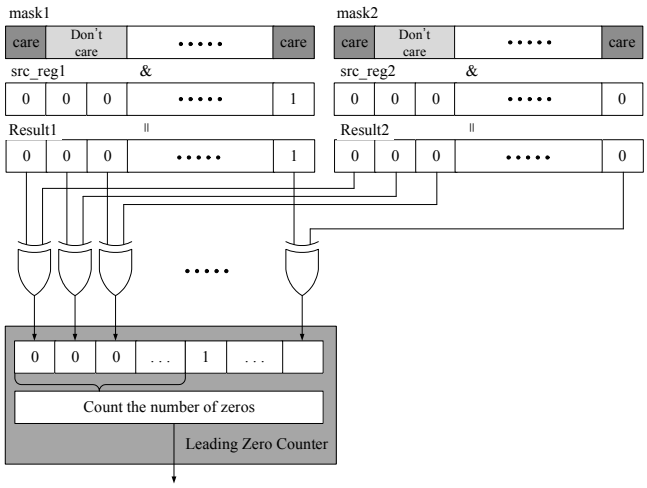

Fig. 5 mcmpclz instruction

2.2.2 Implementation

Table 2 shows the total operation cycles of the general processor running the NX operations without the proposed NX instructions. The clz instruction for example, due to the fact that this instruction is not provided on a general processor, can run on 32 cycles maximum per one clz in a worst case within a 32-bit width register. So as to implement the clz operation we have designed the specific clz instruction by using LISA(Language for Instruction Set Architecture) [4] and have added it on the NX extension module. By the Coware Design compiler, the NX extended processor model has been derived as a synthisalbe Verilog-HDL model. But due to the non-efficient coding of the automatically derived Verilog-HDL model, some HDL optimizations were done by hand coding so as to satisfy the required operation speed with the area of the network application. Therefore, to optimize the functional units of the proposed instruction sets, it was replaced with a fully optimized hand-written verilog-HDL. Fig. 6 shows the optimized functional units of the proposed instruction sets.

Table 2. NX insturctions

NX Instruction set group	Instruction	Description
set	sub(3), shift(2), or(1)	6
clear	sub(3), shift(2), not(1), and(1)	7
test	sub(3), shift(2), cmp(1)	6
flip	sub(5), shift(11), not(1), or(2)	19
cmp	cmp(1)	1
clz	clz(1)	1
cmpclz	cmp(1), clz(1)	2
mcmp	cmp(2), clz(1)	3
mcmpclz	cmp(2), xor(1), clz(1)	4

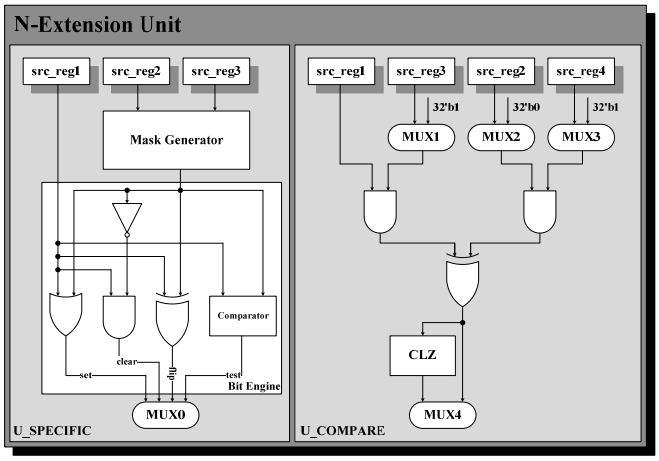

Fig. 6 NX Function Unit

- **U_SPECIFIC**

The U_SPECIFIC has a 'Mask generator' which masks a specific region to '1' and '0' in 32-bit data with the src_reg2 and src_reg3. The value of the Mask generator and the src_reg1 is sent to the 'Bit engine' and four different SPECIFIC instructions are operated. The result is selected by the lower 2-bit in the function code and is sent to the EX/MEM pipeline register.

- **U_COMPARE**

At the U_COMPARE, the [2:1] bit of the function code is sent to the MUX2 as the control signal. If the [2:1] bit of the function code is all '0', it represents a clz instruction, and in other cases the data of the src_reg2 is sent to the result. The [2] bit of the function code is sent to the MUX1 and MUX3 as a control signals, and if the signal is '1' it send the src_reg3 and the src_reg4 to the result, or it send 32'b1 data to the result in cases where the [2] bit is '0'. The [0] bit of the function code is used as a control signal for MUX4, and this determines a CLZ block operation when the value is '1'. The CLZ block was designed using the Leading One Detector on [5] so as to decrease the throughput.

3. Experiment and Result

To evaluate the performance of the proposed network ASIP, we have used three ASIP models. The first ASIP model is a general basic ASIP without the NX instruction sets. The second is the ASIP model with the N-Extension functional units, but is without the NX optimization. Last, the third model is the optimized model of the second model.
Table 3 shows the synthesis results by using the Dong-Bu 0.18um CMOS library using the Synopsys Design Compiler. The non-optimized model had a 26.8% area increase compared to the original basic ASIP, and the optimized model had a 25.7% area increase compared to the original basic ASIP.

Table 3. Total area of each evaluation models (Eq.NAND2x1)

ASIP model	Basic	Non-optimize	Optimize
Total area [gate] (Normalized)	80840.003 (1)	102526.336 (1.268)	101616.337 (1.257)

Table 4. Area and the operating speed of functional units in EX stage

Area[gate] (time[ns])	Basic	Non-optimize	Optimize NX
U_ALU0	2076.333 (1.89)		
U_ALU1	405.666 (2.27)		
U_MULT	8758.666 (3.76)		
U_SHFT	3363.333 (2.37)		
U_COMPARE	0	763.999 (3.35)	379.666 (2.55)
U_SPECIFIC	0	4520.000 (3.75)	3312.346 (2.99)

Table 4 shows the area and the operating speed of each functional unit on the EX stage. As the critical path of the 5-stage pipeline laid on the EX stage, the operating speed of the EX-stage determines the total operating speed of the processor. As the result of each unit synthesis, the U_MULT block had the slowest operating speed. The maximum operating speed had about 156.Mhz(6.38ns). For optimization of the processor, there was no need to optimize the U_COMPARE and the U_SPECIFIC unit, but it is best to optimize these for the cases where the processor used in the router does not use the multiplication unit. So it was considered best to optimize the two new units. As a result of the optimization, the operation speed increased about 23.9%, 20.3% and the area decreased about 50.3%, 26.7% for the U_COMPARE and the U_SPECIFIC respectively.

4. Conclusion

In this paper, we proposed new instruction sets to design a new network ASIP. The proposed instruction functional unit block, U_COMPARE and U_SPECIFIC, had respective operating speeds of 2.55ns and 2.99ns with the equivalent gate of 379.666 and 3312.346, when synthesized by using the Dong-Bu 0.18 CMOS library. By using the proposed instruction sets, it can operate the packet processing with fewer instructions than a general processor. Due to the decreased operating packet processing code size and the operating speed, it can be seen as an appropriate hardware accelerator for the router.

7. Acknowledgments

This work was supported by ETRI (Electronics and Telecommunications Research Institute) from the Research Program of multimedia convergence network on chip technology. EDA tools which were used in this work were supported by the IC Design Education Center (IDEC).

References

[1]R. M. Hinden, "IP Next Generation Overview," Commun. ACM, vol. 39, no. 6, 1996, pp. 61–71.
[2]Scott Weber and Liang Cheng, "A Survey of Anycast in IPv6 Networks," Communications Magazine, IEEE, Vol.42, Jan 2004, pp 127- 132.
[3]Ha-young Jeong, Hyoung-pyo Lee, and Yong-surk Lee, "A Low-cost Multimedia ASIP Architecture for H.264/AVC," The 22nd International Technical Conference on Circuits/Systems, Computers and Communications(ITC-CSCC 2007), Vol.2, July 2007, pp. 777-778.
[4]Andreas Hoffmann, Tim Kogel, Achim Nohl, etc. "A Novel Methodology for the Design of Application-Specific Instruction-Set Processors(ASIPs) Using a Machine Description Language," IEEE Transactions on Computer-AIDED design of intergrated circuits and systems, Vol.20, No 11, November 2001.
[5]H. Suzuki, et. Al, "Leading-Zero Anticipatory Logic for High-Speed Floating Point Addition," IEEE Journal of Solid-State Curcuits, vol. 31, No. 8, August 1996.