ELectronics EXpress

IEICE Electronics Express, Vol.3, No.8, 149–155

Studies on the accuracy of numerical operations with embedded CPUs

Satoshi Kawamura[1a)], **Takahiro Nakanishi**[2], **Hitoaki Yoshida**[2], **Kazuaki Oozeki**[3], **Kazuo Fujimaki**[3] **and Ryuji Gotoh**[3]

[1] *Faculty of Science and Engineering, Ishinomaki Senshu University*

1 Shinmito, Minamisakai, Ishinomaki, Miyagi 986–8580, Japan

[2] *Super Computing and Information Sciences Center, Iwate University*

4–3–5 Morioka, Iwate 020–8550, Japan

[3] *R&D Center, ADTEK System Science Co., Ltd.*

1–145–1 Nimaibashi, Hanamaki, Iwate 025–0312, Japan

a) *kawamura@isenshu-u.ac.jp*

Abstract: In this study, the authors performed a computer experiment on Renesas SH4 and Intel PXA255, which are very popular CPUs for embedded use, to verify their floating-point operation performance in addition (subtraction) and multiplication. The experimental results show that outputs of the embedded CPUs have possible errors. The errors have been successfully removed by clearing the low-order 10 bits to zero in the field of fractional bits of the floating point numbers. The result suggests that special care must be used in the accurate calculation when using an embedded CPU for floating point operations.

Keywords: embedded CPU, capability of numerical operation, computational error, operation accuracy

Classification: Science and engineering for electronics

References

[1] Embedded Linux industry trend ∼What are the advantage and the weak point of Embedded Linux?∼, Development of Embedded Systems Forum Web site, http://www.atmarkit.co.jp/fembedded/trend0411/trend01.html (in Japanese)

[2] SH-4 Programming Manual, Web Site. http://documentation.renesas.com/eng/products/mpumcu/e602156_sh4.pdf

[3] Intel® PXA255 Applications Processors Developer's Manual, ftp site. ftp://download.intel.com/design/pca/applicationsprocessors/manuals/27869302.pdf

[4] D. A. Patterson, J. L. Hennessy, P. J. Ashenden, and J. R. Larus, *Computer Organization and Design: The Hardware/Software Interface (The Computer Architecture and Design Series)*, Morgan Kaufmann Pub., 2004.

[5] J. H. Wilkinson, "*Rounding errors in algebraic processes*," Prentice-Hall, 1963.

[6] MISRAC Web Site. http://www.misra-c2.com/

DOI: 10.1587/elex.3.149
Received February 13, 2006
Accepted March 27, 2006
Published April 25, 2006

[7] IA-32 Intel Architecture Software Developer's Manual Volume 1: Basic Architecture, ftp Site. ftp://download.intel.com/design/Pentium4/manuals/25366517.pdf
[8] AMD64 Architecture Programmer's Manual Volume 1: Application Programming, Web Site. http://www.amd.com/us-en/assets/content_type/white_papers_and_tech_docs/24592.pdf
[9] S. Kawamura, H. Yoshida, M. Miura, M. Saito, T. Takahashi, and K. Oozeki, "CPLD Based Implementation of Artificial Neuron Using Nonlinear Activation Function," *Proc. of the 17th Digital Signal Processing Symp. (CD-ROM)*, B7–3, 2002. (in Japanese)

1 Introduction

Severe restrictions on costs and operating environments are limiting the capabilities of embedded CPUs. General-purpose CPUs used in PC run at speeds from hundreds of MHz to several GHz, but embedded CPUs (also called MPUs) do not run faster than about tens of MHz to hundreds of MHz. Regarding numerical operations of CPU, general-purpose CPUs used in PC, have highly efficient functions and performance of Floating point number Processing Units (FPUs), but embedded CPUs often have no or only a few limited functions [1, 2, 3].

With the advance of ubiquitous computing and the spread of systems with embedded CPUs, such as cellular phones, home information appliances, and other CPU-embedded devices, processing configurations are increasing in these devices for image processing, cryptographic applications, and other comparatively processing. For the adaptive processing with embedded systems, an artificial intelligence technique is used. Generally, high operation accuracy and the double data type of floating point expressions is needed for speech and image signal processing and neural networks, and so on.

A CPU-embedded device executes processing within the capabilities of its CPU and memory. Embedded CPUs are enhanced especially in their operational capabilities for specific applications and functions. As compared with general-purpose CPUs for PCs, embedded CPUs are often restricted in operating speed, operation accuracy, and function [1, 2, 3]. In particular, the capability of the floating point operation is limited in almost all cases.

When constructing an actual system, basic knowledge about the employed devices is important. Floating point arithmetic systems and their behavior have been theoretically studied [4, 5] and the method of programming for embedded systems has been standardized [6]. For a comparison of the operational accuracy and computational results between general-purpose CPUs and embedded CPUs, however, no information is available from manufacturers' technical documents [2, 3, 7, 8] and no detailed studies have been made in a real system (device).

This paper reports the results of evaluating Renesas SH4 [2] and Intel PXA255 [3], which are very popular CPUs for embedded use, to verify their floating-point operation performance, especially in addition (subtraction) and

DOI: 10.1587/elex.3.149
Received February 13, 2006
Accepted March 27, 2006
Published April 25, 2006

multiplication. For floating point expressions, the double data type (64 bits) of the C language was used. For a comparison of the computational results, the most popular x86 compatible CPUs (IA-32 and x86-64) were used. The computer experiment clarified that some systems could cause errors even in the same processing. This paper also reports that clearing the low-order 10 bits to zero in the field of fractional bits (rounding) made the computational results identical. Note that the results cannot be estimated from technical documents of each CPU [2, 3, 7, 8].

This paper is structured as follows. Section 2 introduces the method of the computer experiment to compare the capability of numerical operations between CPUs. Section 3 presents and discusses the results of the computer experiment, and finally the conclusions are presented in Section 4.

2 Computer experiment

To verify various CPUs for their accuracy of numerical operation, the authors compared the results of the same operations. For this verification, floating point operations were used because they are particularly straightforward for showing differences (due to the difference of implementation of numerical operations). For the operations, the double data type of the C language was used. The length of the double data type is 64 bits (sizeof( ) function returns 8 bytes) in all systems. In manufacturers' technical documents of each CPU, it is stated that operations conforming to IEEE 754 are possible but does not mention errors clearly [2, 3, 7, 8].

2.1 Processing used to compare the computational error

Eq. (1) shows the object of computation used in the experiment. This expresses Eq. (2) of a general neuron model having piecewise nonlinear output function without using the exp function or division [9].

$$\begin{cases} f(x) = 1.0, x \geq 12.0 \\ f(x) = -0.000337592 \times (x-12)^2 + 1.0, 12.0 > x \geq 3.75 \\ f(x) = -0.0324362 \times (x-3.83587)^2 + 0.977262, 3.75 > x \geq 0.0 \\ f(x) = 0.0324362 \times (x+3.83587)^2 + 0.022738, 0.0 > x \geq -3.75 \\ f(x) = 0.000337592 \times (x+12)^2 + 0.0, -12 > x \geq -3.75 \\ f(x) = 0.0, x \leq -12.0 \end{cases} \tag{1}$$

$$f(x) = \frac{1}{1+\exp(-u)}, u = \sum_{i=1}^{n} w_i x_i \tag{2}$$

2.2 Environment used for comparison

This section describes the environment used to compare the operational accuracy. Table I shows the environment used for the computer experiment.

The x86 and its compatible CPUs (AMD Duron, Intel Xeon), which are widely used for general PCs, AMD Opteron (32-bit mode, gcc -m32) for servers, and Renesas SH4 and Intel PXA255 (ARM Architecture v.5TE),

DOI: 10.1587/elex.3.149
Received February 13, 2006
Accepted March 27, 2006
Published April 25, 2006

Table I. Environments used for comparison: CPU and OS information

(For Opteron CPU, 32-bit codes were generated by gcc -m32 compilation.)

		AMD Duron	AMD Opteron	Intel Xeon	Renesas SH4	Intel PXA255
CPU*1	NAME	AMD Duron™ Processor	AMD Opteron™ Processor 240	Intel Xeon™ CPU 2.40GHz	Renesas SH4	Intel® PXA255 Processor
	Frequency	1194.931MHz	1403.219MHz	2392.464MHz	266.81MHz	400MHz
	Details	cpu family : 6 model : 7 stepping : 1	cpu family :15 model : 5 stepping : 1	cpu family :15 model : 2 stepping : 5	Machine7751R Julian	Intel Xscale (ARM v.5TE)
	FPU	Yes	Yes	Yes	Yes	No
OS *2		Turbolinux Workstation 7.0 (Monza)	Turbolinux AMD64 8.0 (Fregrance Moon)	SuSE Linux 9.1 (i586)	testing/unstable (Debian Linux 2.4.21)	Microsoft PocketPC Version 4.20.1031 (Build 13100) Windows Mobile™ 2003 software for Pocket PC
Compiler and development environment *3		gcc version 2.95.3 20010315(release)	gcc version 3.2.2 (Turbolinux 3.2.2-5)	gcc version 3.3.3 (SuSE Linux)	gcc version 3.0.4	Microsoft eMbedded Visual C++ 4.00.1631.0 (SP4)

*1 CPU information: /proc/cpuinfo for GNU/Linux
*2 OS information: Release information under /etc for GNU/Linux
*3 Compiler information: gcc version for GNU/Linux

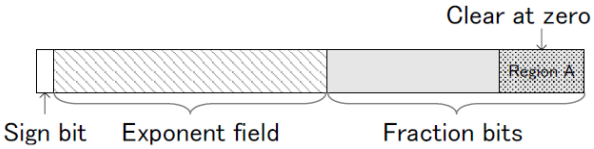

Fig. 1. Floating-point expression and zero clearing method (Clearing the end of the field of fractions bits to zero)

which are mainly implemented for embedded use, were verified. All are 32-bit CPUs of little-endian numeral systems (Opteron is a 64-bit CPU but was used in the 32-bit mode). Of these CPUs, only Intel PXA255 does not have a floating-point operation function and uses a software library for its floating point operations.

For Intel PXA255 in the computer experiment, a Pocket PC was used as the OS and Microsoft EMbedded Visual C++ (SP4) was used as the compiler. For other CPUs, the OS was GNU/Linux and the compiler was gcc (GNU C Compiler).

Because floating point operations are used in this experiment, errors in the computational results may occur. Therefore, Eq. (1) was employed as the object of computation and the function output $f(x)$ was obtained by varying the input value x from -20.0 to 20.0 in units of 0.01. Because computational errors were anticipated, the end (Region A in Fig. 1: low-order bits of the field of fractional bits) of the output value of each operation (addition (subtraction) or multiplication) was cleared to zero, as shown in Fig. 1. This zero clearance (rounding) was also applied to input value x.

DOI: 10.1587/elex.3.149
Received February 13, 2006
Accepted March 27, 2006
Published April 25, 2006

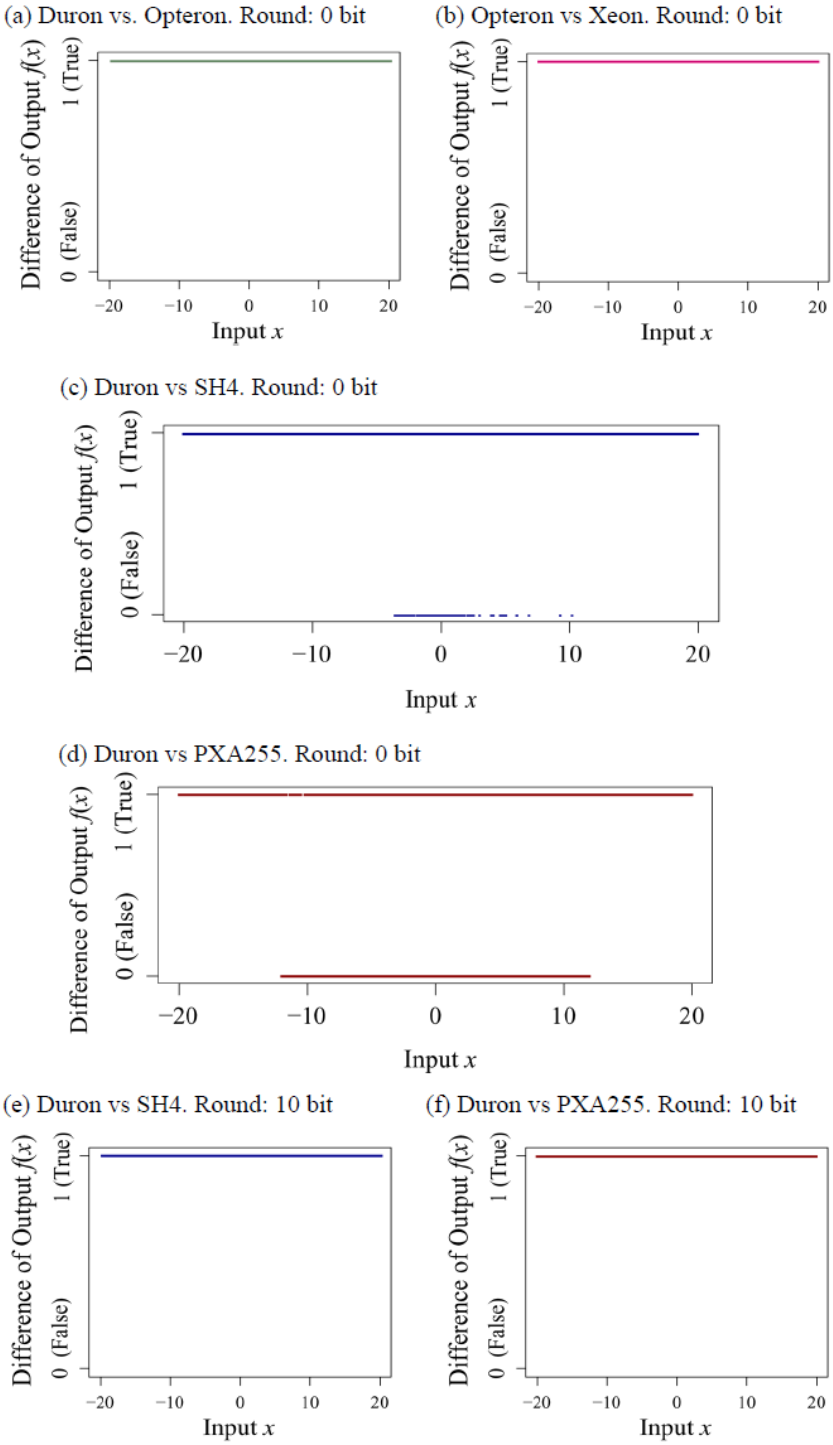

Fig. 2. Computational results of different CPUs - 0 (False) for any difference and 1 (True) for no difference. (a) to (d): After no zero clearance (rounding), (e) and (f): After 10-bit zero clearance (rounding).

3 Results and discussion

Fig. 2 shows the following results of the computer experiment: (False) for any difference in the computational results and 1 (True) for no difference. For x86 and its compatible CPUs (AMD Duron, AMD Opteron (32-bit mode),

DOI: 10.1587/elex.3.149
Received February 13, 2006
Accepted March 27, 2006
Published April 25, 2006

and Intel Xeon), all output values were identical with errors even when the computational results were not cleared to zero (rounded) (see Fig. 2 (a) and (b)). Therefore, this paper compares Renesas SH4 and Intel PXA255 with the AMD Duron.

Fig. 2 (c) and (d) compare embedded CPUs Renesas SH4 and Intel PXA255 with AMD Duron when zero clearance was not applied. In the figures, 0 (False) for Renesas SH4 and Intel PXA255 is seen in many areas, meaning that there was a difference in the computational results. In particular, PXA255 with no FPU output shows different computational results from AMD Duron in a wide range (area) of input values x. This may be because the OS and compiler are different in this environment.

Because the same operation ends in different results, the technique shown in Fig. 1 was applied to clear the low-order bits of the output values to zero. As more bits were cleared to zero, the area of difference in the computational results became narrower. When 10 bits were cleared to zero in the end, there were no more differences in the computational results. Fig. 2 (e) and (f) show the computational results with the 10-bit zero clearance. Although not shown in Fig. 2, the output values from x86 and its compatible CPUs (AMD Duron, AMD Opteron in the 32-bit mode, and Intel Xeon) were identical, even when zero clearance was used.

From the computer experiment results for the output of the neuron nonlinear approximate function in Eq. (1), it became clear that 10-bit zero clearance would be needed to make the computational results identical among all CPUs tested. This cannot be estimated from technical documents of each CPU [2, 3, 7, 8].

4 Conclusion

On several CPUs popular for embedded use, a computer experiment was performed for floating-point operation performance based on a general neuron model having piecewise nonlinear output function. The floating point operations here were addition (subtraction) and multiplication. In the experiment, the double data type of the C language was used. The length of the double data type is 64 bits (sizeof( ) function returns 8 bytes) in all systems. The x86 CPU and its compatible CPUs output no computational differences in the computational results, regardless of whether zero clearance (rounding) was applied to the low-order bits in the field of fractional bits of the output value. In contrast, the embedded CPUs Renesas SH4 and Intel PXA255 output greatly different computational results. In particular, Intel PXA255 with no FPU output different results from other CPUs in a wide range (area) of input values x.

These results clarified that embedded CPUs would output different computational results for the x86 and its compatible CPUs used for comparison. It also became clear that the embedded CPUs (Renesas SH4 and Intel PXA255) used this time could absorb the computational errors and output identical results by clearing the low-order 10 bits to zero (rounding) in the

DOI: 10.1587/elex.3.149
Received February 13, 2006
Accepted March 27, 2006
Published April 25, 2006

field of fractional bits of the floating point numbers in the same output.

Because a similar tendency is also anticipated in practical applications using embedded CPUs, extreme care must be taken for an accurate calculation when using an embedded CPU for floating point operations.

Acknowledgments

This work was supported by Dreamland Iwate Strategic Research Promotion Project from Science and Technology Division of Iwate Prefecture.

DOI: 10.1587/elex.3.149
Received February 13, 2006
Accepted March 27, 2006
Published April 25, 2006