A Binary Floating-Point Adder with the Signed-Digit Number Arithmetic

Shugang Wei
Gunma University
Department of Computer Science
Tenjin-cho 1-5-1, Kiryu-shi, Gunma 376-8515
Japan

Abstract: In a conventional binary floating-point number arithmetic system, two's complement number representation is often used to perform addition/subtraction in a floating-point adder. Since the significand of an addition operand is usually expressed as a sign-magnitude number representation, the swapping operation of two operands and the carry propagation in the addition will limit the arithmetic speed. In this paper, we introduce a radix-two signed-digit(SD) number arithmetic to the floating-point number arithmetic. Then the swapping operation is not required and the carry propagation becomes free for the inner addition. We present an addition circuit architecture using the SD arithmetic with a normal binary floating-point number representation in both input and output. Efficient SD-binary conversion and normalization circuits are also proposed.

Key–Words: Floating-point number, Signed-digit number, Arithmetic circuit, Binary addition, IEEE standard 754.

1 Introduction

Since fixed-point number systems have certain limitations in the relatively small range of numbers that can be represented and the possible loss of significant digits during computations, floating-point number systems are often used to manipulate very large and very small numbers for scientific computation and digital signal processing[1, 2].

A floating-point number is usually represented as an exponent part and a significand part, so that the arithmetic with floating-point numbers is very complicated comparing to that with the fixed-point numbers. In basic arithmetic operations, the floating-point addition and subtraction are most complicated ones that consist of the operations such as exponent manipulation, alignment shifting, addition, normalization and rounding. A number of adders for the IEEE Standard 754, a binary floating-point data format, have been presented[1, 3]. However, the swapping operation of two operands in the ordinary binary number representation and the carry propagation during the addition of them will limit the arithmetic operation speed.

It is well known that the carry propagation is limited to one position during additions of signed-digit(SD) numbers[4]. The redundant SD number representation is often used to achieve high-speed arithmetic[6]. In this paper, we present a new architecture of the floating-point adder by introducing a radix-two SD number arithmetic. Therefore, the swapping operation of two operands is not required and the carry propagation becomes free for the inner addition. We also present efficient SD-binary conversion and normalization circuits. The design and simulation results show that the delay time of the proposed floating-point adder is reduced to 77 % comparing to the conventional one.

2 Addition in Floating-Point Number System

2.1 IEEE Floating-Point Number System

The floating-point number system has a very large range of number representations, comparing to the fixed-point number system with the same wordlength. Figure 1 shows a 32-bit floating-point data format, which is well known as " IEEE standard 754"[4]. The sign bit is the leading bit and the 8-bit exponent field precedes the 23-bit significands field. The exponent is biased by adding 2^7-1 and the significand is in a sign-and-magnitude notation. The 1 is hidden in the number representation, so that the value range expressed by the significands field is $[1, 2)$. Therefore, the value of such a floating-point number x is given by the following equation:

$$x = (-1)^s M \times b^{e-127}, \qquad (1)$$

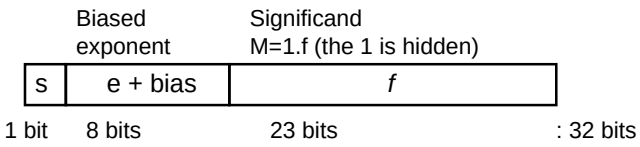

Figure 1: IEEE standard floating-point number representation.

where $(-1)^s$ denotes the sign of x. That is, $s = 0$ for positive numbers and $s = 1$ for negative numbers. $b = 2$ is the exponent base, and e expresses the exponent. $M = 1.f$ and f is a pure unsigned fractional number. Most of addition/subtraction time is for the arithmetic operations on the significands.

2.2 Conventional Adder Architecture

Let x_1 and x_2 be two floating-point numbers, as shown in Fig.1. Since the subtraction can be performed by $x_1 - x_2 = x_1 + (-x_2)$, we consider the addition without loss of generality. When $e_1 \geq e_2$, the addition can be performed as follows:

$$x_1 + x_2 = ((-1)^{s_1} M_1 + (-1)^{s_2} M_2 2^{-(e_1 - e_2)}) 2^{e_1 - bias}$$

To implement the above arithmetic, the following steps are required for adding two floating-point numbers:
(1) Calculate the difference of the two exponents,

$$d = e_1 - e_2.$$

(2) Shift the significand of the smaller number by d bit positions to the right.
(3) Add the aligned significands and set the exponent of the result equal to the exponent of the larger operand.
(4) Normalize the resultant significand and adjust the resultant exponent if necessary.

When $x_1 < x_2$ and $e_1 = e_2$ specially, the swapping of the operands with a comparison is required, and a pretty long delay time for the comparison is necessary.

A conventional architecture of floating-point adder is illustrated in Fig.2[4, 7]. The swapper compares the magnitudes and changes the order of the significands. The bit-inverter makes the two's complement for the negative value of the second operand. Then the aligned binary numbers are added in the binary adder. The rounding and shifting operations are performed for the normalization. In the floating-point addition circuit, swapping two unsigned significands with a comparison time and adding them with a carry propagation delay time will limit the arithmetic operation speed.

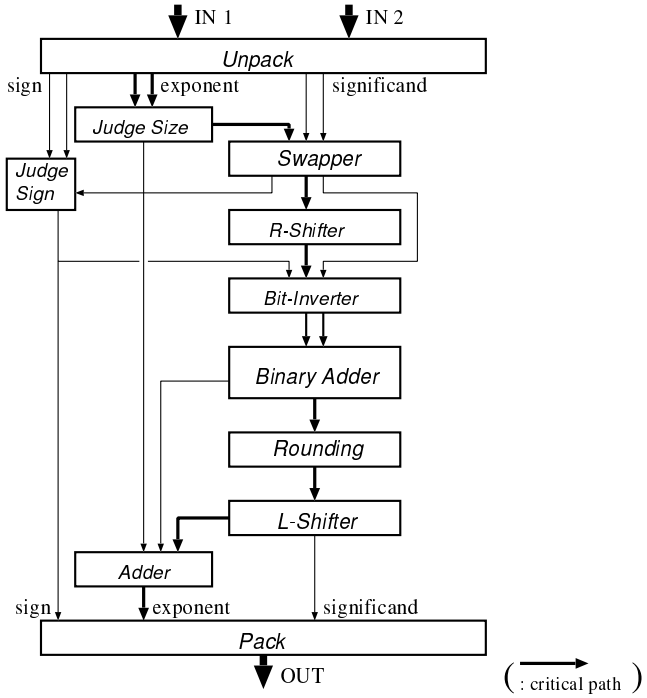

Figure 2: Conventional binary floating-point adder.

3 Floating-Point Adder Using SD Arithmetic

3.1 Architecture of the Proposed Floating-Point Adder

We introduce a radix-two SD number arithmetic to the floating-point addition for getting a shorter delay time. An architecture of the floating-point adder using the SD number arithmetic is shown in Fig.3. The swapping is not required for the addition of two SD numbers, then the unsigned significands are directly aligned by shifting in R-shifters. The shift bit number to right is determined by the values of the exponents in the circuit **Judge Size**. Then the guard digits, R and S, are generated for rounding. After attaching the signs to the addends in **SD Coder**, the SD addition of two SD numbers are performed. The carry propagation is free for the SD addition, so that a high-speed floating-point adder may be designed. If the operands have the same sign, the exponent is increased by 1 and the SD number is shifted to right. To convert the SD number into the IEEE standard floating-point data format, an **SD Decoder** is used to transfer the SD number into a binary one, and a properly left-shifting is performed for the postnormalization. For the SD-to-binary conversion, a binary addition is performed and the carry propagation will arises for the binary number arithmetic. In parallel, the number of shift bit number,

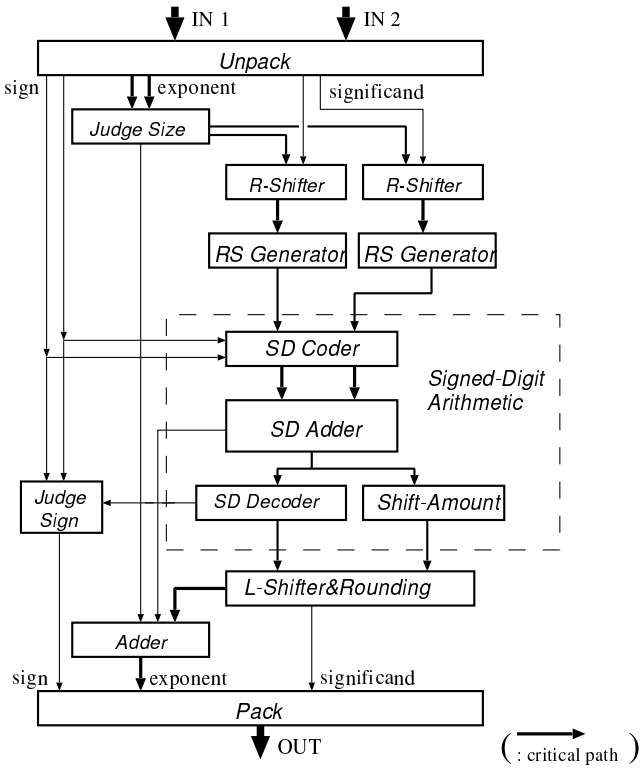

Figure 3: Architecture of floating-point adder using SD arithmetic.

which is required in the following **L-shifter**, are decided by the circuit **Shift-Amount**.

When the operands have the same sign, the sign will be as the output sign; when they have different signs, the output data sign will be determined by the calculation result in **SD Decoder**.

3.2 Sign-Digit Arithmetic

A number x can be represented by a p-digit radix-two SD number representation as follows:

$$\begin{aligned} x &= x_{p-1}2^{p-1} + x_{p-2}2^{p-2} + \cdots + x_0, \\ & x_i \in \{-1, 0, 1\} \ (i = 0, 1, \cdots, p-1), \end{aligned} \quad (2)$$

which can be denoted as $x = (x_{p-1}, x_{p-2}, \cdots, x_0)_{SD}$. The SD number representation has redundancy; for example, 7 may be represented by $(0, 1, 1, 1)_{SD}$, $(1, -1, 1, 1)_{SD}$ or $(1, 0, 0, -1)_{SD}$ for $p = 4$. By using the redundant number representation, parallel arithmetic can be achieved without the carry propagation which occurs during addition in an ordinary binary system. In the SD number representation, x has a value in the range of $[-(2^p - 1), 2^p - 1]$. An SD adder is shown in Fig.4, performing the following two steps.

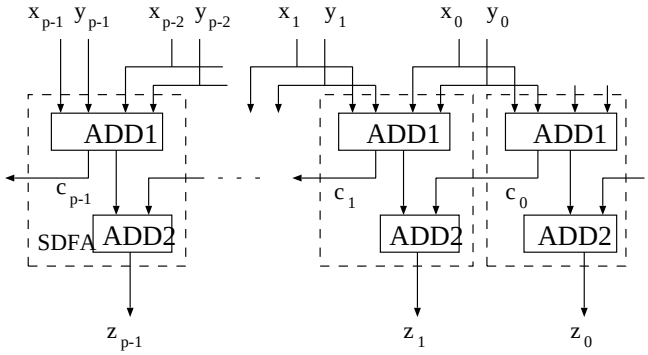

Figure 4: Block diagram of SD adder.

SD Addition $(x + y)$**:** Let c_i, w_i and z_i be the intermediate carry, intermediate sum and sum at the ith SD digit ($i = 0, 1, \cdots, p-1$), respectively. For each digit, the following two steps are performed.
Add1: When $abs(x_i) = abs(y_i)$,

$$(w_i, c_i) = (0, (x_i + y_i)\ div\ 2);$$

when $abs(x_i) \neq abs(y_i)$,

$$(w_i, c_i) = \begin{cases} (-(x_i + y_i), x_i + y_i) & \text{if } (x_i + y_i) \text{ and } (x_{i-1} + y_{i-1}) \text{ have the same sign} \\ (x_i + y_i, 0) & \text{otherwise} \end{cases}$$

Since $x_i, y_i \in \{-1, 0, 1\}$, $w_i, c_i \in \{-1, 0, 1\}$.
Add2: $z_i = w_i + c_{i-1}$. Thus,

$$z = z_p 2^p + z_{p-1}2^{p-1} + \cdots + z_0$$

It is always true that $2c_i + w_i = x_i + y_i$, and w_i and c_{i-1} do not have the same sign so that $z_i \in \{-1, 0, 1\}$. Thus the carry propagation is limited to one digit position in an SD adder as shown in Fig.4..

An SD number obtained from the SD addition should be converted into a binary number, then the rounding and the normalization is performed. For the postnormalization, the shift bit number is required. We can convert the SD number into the binary and calculate the number of shift bits in parallel. In **SD Decoder**, at first the SD number is divided into a positive and a negative binary number. Then an addition in the binary number representation is performed, where the negative number is represented by a two's complement. If the sign is negative, then the number is changed by making the complement at every bit.

The shift bit number is obtained directly from the SD number in parallel with the SD-to-binary conversion. A method similar to a Leading Zero circuit is applied, performing the following procedure.
Shift Amount: Consider a p-digit SD number, $x = (x_{p-1}, x_{p-2}, \cdots, x_0)_{SD}$. Let t be the shift bit number.

Table 1: Example of the proposed floating-point addition.

R-Shifting:		
Sig A	1.01000001	R:0,S:0
Sig B aligned	0.01110001	R:1,S:1
SD Coding and	**ADD1:**	
Sig A_{SD}	1.01000001	R:0,S:0
Sig B_{SD} aligned	$0.0\bar{1}\bar{1}\bar{1}0001\bar{1}$	R:$\bar{1}$,S :$\bar{1}$
ADD2: c_i	$00.0\bar{1}000001\bar{1}$	0
w_i	$1.001\bar{1}0000$	$1\bar{1}$
Sig A+B	$1.0\bar{1}\bar{1}\bar{1}0001\bar{1}$	$1\bar{1}$
SD Recoding:		
Positive(+)	01.00100000	10
Negative(-)	11.10101110	11
Sig $(A+B)$	00.11001111	01
Normalizatioin:		
Sig $(A+B)_N$	1.10011111	Shift 1 bit

```
t := 0
for i = p-1 downto 0,
begin
  if x(i) = 0 then t := t + 1
    else if x(i)=x(i-1) <> 0 then exit
        else
        for j = i-1 downto 0,
        if x(i)=-x(j-1)<>0 then t := t
                           else exit
end
```

Shift-Amount circuit based on the above procedure is more complicated than the normal Leading Zero circuit. However, the delay time may be smaller than that of **SD Decoder** circuit.

Example 1: Comsider a simple addition of A and B having the following binary expressions:

$$\begin{cases} A = 2^3 \times 1.01000001 \\ B = -2^1 \times 1.11000111. \end{cases}$$

After shifting and generating the guard digits, R and S, the SD number representations for them can be obtained in **SD Coder**. Table 1 illustrates the arithmetic operation process, where $\bar{1} = -1$. The shift bit number is 1. The round-to-nearest method is used for the normalization and the final addition result is $A + B = 2^2 \times 1.10011111$.

4 On VLSI Implementation

We specify a binary representation for a radix-two signed-digit a_i, that is, $a_i(1)$ is the sign and $a_i(0)$ is the absolute value of a_i. Thus, a p-digit radix-two SD number a is represented by a vector with $2p$-bit length.

$$\begin{aligned} a &= (a_{p-1}, a_{p-2}, \cdots, a_0)_{SD} \\ &= [a_{p-1}(1)a_{p-1}(0)\ a_{p-2}(1)a_{p-2}(0) \\ &\quad \cdots a_0(1)a_0(0)] \end{aligned} \tag{3}$$

For example, $(1, 0, 0, -1)_{SD} = [01000011]$. Using the binary representation, the arithmetic operations in functional blocks including the SD arithmetic circuits are described and simulated. By using VHDL description codes, and the simulation and the logic circuit synthesis are performed by a synthesis software tool. In our experiments, the delay time is obtained by the simulation under the condition of $1\mu m$ CMOS gate array technology.

Table 2: Performance of the proposed floating-point adder

	Area(Gates)		Delay time [ns]	
Main circuits	Binary	SD	Binary	SD
Swapper	118	–	18.93	–
Adder	228	534	26.90	5.93
RS Generator	244	244	8.34	8.34
Rounding	65	–	9.75	–
SD Coder	–	24	–	0.79
SD Decoder	–	208	–	26.91
Shiftamount	–	172	–	(24.39)
Total	945	1449	92.90	71.79

In Table 2, the design results of the floating-point adder having IEEE standard data format is shown, based on the SD arithmetic method. For the performance evaluation, a floating-point adder with a binary arithmetic is also designed for comparing with the presented one. In the proposed floating-point adder, the swapper, which is used in the binary adder, is not required. Moreover, the addition is performed by the SD adder. Therefore, the delay time of the addition circuit is reduced by 77 %, comparing to the binary one. However, the presented floating-point adder needs about 1.5 times of hardware the binary one needs.

5 Conclusion

A new architecture of a floating-point adder using the SD arithmetic has been proposed. Because the swapping operation is not required and the carry propagation is free in the SD addition, a high-speed floating-

point adder can be implemented by the presented method.

The design and simulation results under the condition of $1\mu m$ CMOS gate array technology show that the SD floating-point adder has a shorter delay time by 77% comparing to that of a conversional floating-point adder. Our studies also focus on the error analysis of the proposed adder and the applications.

References:

[1] A. R. Omondi,*Computer Arithmetic Systems: Algorithms, Architecture and Implementation*, Prentice Hall, 1994.

[2] L. Wanhammar,*DSP Integrated Circuits*, Academic Press, 1999.

[3] "Leading-zero Anticipatory Logic for High-speed Floating Point Addition", Technical Report of IEICE DSP95-98 ICD95-147, 1995.

[4] ANSI/IEEE Standard 754-1985 for Binary Floating-point Arithmetic, IEEE, 1985.

[5] A.avizienis,"Signed-digit number representations for fast parallel arithmetic", IRE Trans. Elect. Comput., EC-10, pp.389-400, 1961.

[6] S.Wei and K.Shimizu,"Residue Arithmetic Circuits Using a Signed-Digit Number Representation", IEEE Proc. of ISCAS 2000, pp.I24-I27, May 2000.

[7] Woo-Chan Park, "Design of the Floating-point Adder Supporting the Format Conversion and the Rounding Operations with Simultaneous Rounding Scheme", IEICE Trans.Inf.&Syst.,Vol.E85-D,No.8, 2002.