FLOATING-POINT NUMBER MULTIPLICATION COMPUTATION METHOD AND APPARATUS, AND ARITHMETIC LOGIC UNIT

[0001]

TECHNICAL FIELD

[0002] This application relates to the field of data processing technologies, and in particular, to a floating-point number multiplication computation method and apparatus, and an arithmetic logic unit.

BACKGROUND

[0003] A floating-point number is an important digital format in a computer. The floating-point number in the computer includes three parts: a sign, an exponent, and a mantissa. To meet different requirements of different services for data precision, a processor of the computer usually needs to have a capability of performing a multiplication operation on floating-point numbers with different precision.

[0004] Currently, to meet multiplication requirements for floating-point numbers with different precision, a plurality of independent multipliers are usually designed corresponding to precision requirements. For example, if the processor needs to simultaneously support a half-precision multiplication operation, a single-precision multiplication operation, and a double-precision multiplication operation, at least three independent multipliers need to be designed in the processor, so that the three independent multipliers respectively meet half-precision multiplication, single-precision multiplication, and double-precision multiplication.

[0005] In a process of implementing this application, the conventional technology has the following disadvantages:

[0006] A plurality of multipliers that separately support different precision are independently designed in the processor. When a system only uses a multiplier of one type of the precision to perform computation, a multiplier of a remaining type of precision is in an idle state, which

$A_2 = 2^{EA_2}（a_3 + 2^{-S_2} a_4）$, where A_2 is the FP32 in a standard format; EA_2 is an exponent of A_2; a_3 and a_4 are two FP16s in a non-standard format that are obtained through decomposition; and S_2 is a fixed exponent bias value. For the FP16 in a non-standard format, $S_2 = 14$.

[00112] In addition, a common exponent bias value may be extracted for exponents of all the FP16s in a standard format. Therefore, an FP32 in a standard format may alternatively be represented as follows:

$A_2 = 2^{EA_2 - S_2}（a_3' + a_4'）$, where a_3' and a_4' are two FP16s in a non-standard format that are obtained through decomposition. In the foregoing two representation methods, the FP16s obtained through decomposition have the following relationships: $a_3 = 2^{-S_2} a_3'$, and $a_4 = a_4'$.

[00113] Case 2: In a case in which the first-precision floating-point number is an FP64 and the second-precision floating-point number is an FP32, that the FP64 is decomposed to obtain a plurality of FP32s may have the following cases.

[00114] I. One FP64 is decomposed to obtain three FP32s.

[00115] Currently, composition of an FP64 in a standard format is shown in FIG. 4, and includes a 1-bit (bit) sign, an 11-bit exponent (also referred to as an exponent), and a 52-bit mantissa. In addition, there is an omitted 1-bit integer, and the omitted integer is 1. For an FP64 in a standard format, there are totally 53 bits when an integer and a mantissa are added. For the FP32 in a standard format described above, there are totally 24 bits when an integer and a mantissa are added. If an FP64 in a standard format needs to be decomposed to obtain an FP32 in a standard format, three FP32s in a standard format are required.

[00116] The integer and the mantissa of the FP64 in a standard format may be divided into three parts. A first part includes the integer and the first 23 bits of the mantissa, a second part includes the 24th bit to the 47th bit of the mantissa, and a third part includes the 48th bit to the 52nd bit of the mantissa. The three parts each are represented by an FP32 in a standard format.

[00117] It should be further noted herein that when the 48th bit to the 52nd bit of the mantissa in the third part are represented by an FP32 in a standard format, 18 zeros may be first padded after the 23rd bit of the mantissa, that is, the 48th bit to the 52nd bit of the mantissa and the padded zeros are represented by an FP32 in a standard format.

[00118] Therefore, an FP64 in a standard format may be represented as follows:

$A_3 = 2^{EA_3}（a_5 + a_6 + a_7）$, where A_3 is the FP64 in a standard format; EA_3 is an exponent of A_3; and a_5, a_6, and a_7 are three FP32s in a standard format that are obtained

through decomposition.

[00119] II. One FP64 is decomposed to obtain two FP32s.

[00120] To decrease a quantity of FP32s obtained through decomposition, a current FP32 in a standard format may be adjusted. A mantissa of the FP32 is adjusted to 26 bits, and bit quantities of a sign and an exponent remain unchanged. An adjusted FP32 may be referred to as an FP32 in a non-standard format. In this case, there are totally 27 bits when an integer and a mantissa of the FP32 in a non-standard format are added. Therefore, if a mantissa of an FP64 in a standard format needs to be represented by using an FP32 in a non-standard format, only two FP32s in a non-standard format are required.

[00121] An integer and a mantissa of the FP64 in a standard format are divided into two parts. A first part includes the integer and the first 26 bits of the mantissa, and a second part includes the 27th bit to the 53rd bit. The two parts each are represented by an FP32 in a non-standard format.

[00122] Therefore, an FP64 in a standard format may alternatively be represented as follows: $A_4=2^{EA_4}（a_8+a_9）$, where A_4 is the FP64 in a standard format; EA_4 is an exponent of A_4; and a_8 and a_9 are two FP32s in a non-standard format that are obtained through decomposition.

[00123] Case 3: In a case in which the first-precision floating-point number is an FP64 and the second-precision floating-point number is an FP16, that the FP64 is decomposed to obtain a plurality of FP16s may have the following cases.

[00124] I. One FP64 is decomposed to obtain five FP16s.

[00125] For an FP64 in a standard format, there are totally 53 bits when an integer and a mantissa are added. For the FP32 in a standard format described above, there are totally 24 bits when an integer and a mantissa are added. If an FP64 in a standard format needs to be decomposed to obtain an FP16 in a standard format, five FP16s in a standard format are required.

[00126] The integer and the mantissa of the FP64 in a standard format may be divided into five parts. A first part includes the integer and the first 10 bits of the mantissa, a second part includes the 11st bit to the 21st bit of the mantissa, a third part includes the 22nd bit to the 32nd bit of the mantissa, a fourth part includes the 33rd bit to the 43rd bit of the mantissa, and a fifth part includes the 44th bit to the 52nd bit of the mantissa. The five parts each are represented by an FP64 in a standard format. It should be further noted herein that when the 44th bit to the 52nd bit of the mantissa in the fifth part are represented by an FP16 in a standard format, two zeros may be first padded after the 52nd bit of the mantissa, that is, the 44th bit to the 52nd bit of the mantissa and the padded zeros are represented by an FP16 in a standard format.

[00127] In addition, an exponent range of the FP16 is from –15 to 15, which may indicate that a decimal point can move 15 bits to the left and can move 15 bits to the right. When the FP16 in a

point numbers: A1 and A0, and B1 and B0. Four combinations may be obtained for A1, A0, B1, and B0, and there are four second-precision multipliers. Each combination of second-precision floating-point numbers is input into a second-precision multiplier, that is, each combination corresponds to a second-precision multiplier.

[00148] When the quantity of second-precision multipliers is different from the quantity of combinations of second-precision floating-point numbers, as shown in FIG. 6, two first-precision floating-point numbers A and B each are decomposed to obtain two second-precision floating-point numbers: A1 and A0, and B1 and B0. Four combinations may be obtained for A1, A0, B1, and B0, and there is only one second-precision multiplier. In this case, the four combinations of second-precision floating-point numbers are sequentially input into the second-precision multiplier.

[00149] Step 105: Determine a computation result for the plurality of to-be-computed first-precision floating-point numbers based on the intermediate computation result corresponding to each combination.

[00150] In implementation, when the first-precision floating-point number is decomposed to obtain the second-precision floating-point number, an exponent bias value corresponding to each second-precision floating-point number may be further obtained. In several cases in which the to-be-computed first-precision floating-point number is decomposed in step 102, exponent bias values corresponding to the second-precision floating-point number are separately described below.

[00151] Case 1: The first-precision floating-point number is an FP32, and the second-precision floating-point number is an FP16.

[00152] I. One FP32 is decomposed to obtain three FP16s.

[00153] In this case, the FP32 may be represented as follows: $A_1 = 2^{EA_1}（a_0 + 2^{-S_1}a_1 + 2^{-2S_1}a_2）$. Therefore, an exponent bias value corresponding to a_0 is EA_1, an exponent bias value corresponding to a_1 is $EA_1 - S_1$, and an exponent bias value corresponding to a_2 is $EA_1 - 2S_1$. Alternatively, the FP32 may be represented as follows: $A_1 = 2^{EA_1 - S_1}（a_0' + a_1' + a_2'）$. Therefore, exponent bias values corresponding to a_0', a_1', and a_2' each are $EA_1 - S_1$.

[00154] II. One FP32 is decomposed to obtain two FP16s.

[00155] In this case, the FP32 may be represented as follows: $A_2 = 2^{EA_2}（a_3 + 2^{-S_2}a_4）$. Therefore, an exponent bias value corresponding to a_3 is EA_2, and an exponent bias value corresponding to a_4 is $EA_2 - S_2$. Alternatively, the FP32 may be represented as follows: $A_2 = 2^{EA_2 - S_2}（a_3' + a_4'）$. Therefore, exponent bias values corresponding to a_3' and a_4' each are

[00167] When the exponent of the intermediate computation result is adjusted, the exponent of the intermediate computation result corresponding to each combination of second-precision floating-point numbers and the exponent bias value corresponding to the second-precision floating-point number in each combination may be added to obtain the adjusted intermediate computation result.

[00168] In a possible implementation, a format of a second-precision intermediate computation result that is output by the second-precision multiplier may be adjusted to finally obtain a computation result with higher precision. Corresponding processing may be as follows: The second-precision floating-point numbers in each combination are input into the second-precision multiplier to obtain a first-precision intermediate computation result corresponding to each combination of second-precision floating-point numbers, and format conversion is performed on each first-precision intermediate computation result to obtain a third-precision intermediate computation result corresponding to each combination of second-precision floating-point numbers, where precision of the third-precision intermediate computation result is higher than precision of the first-precision intermediate computation result. An exponent of the third-precision intermediate computation result corresponding to each combination of second-precision floating-point numbers is adjusted based on the exponent bias value corresponding to the second-precision floating-point number in each combination, to obtain an adjusted third-precision intermediate computation result. A summation operation is performed on the adjusted third-precision intermediate computation results corresponding to all the groups of second-precision floating-point numbers, to obtain a third-precision computation result for the plurality of first-precision floating-point numbers.

[00169] When format conversion is performed on the first-precision intermediate computation result, zero padding processing may be performed on an exponent and a mantissa of each first-precision intermediate result to obtain the third-precision intermediate computation result corresponding to each combination of second-precision floating-point numbers.

[00170] For example, if the first-precision floating-point number is an FP32 and the second-precision floating-point number is an FP16, the first-precision intermediate computation result that is output by the second-precision multiplier is also an FP32. A format of the first-precision intermediate computation result may be adjusted to the third-precision intermediate computation result, and the third-precision intermediate computation result may be an FP64. To be specific, three zeros are padded after an end bit of an exponent of the first-precision intermediate computation result to extend a quantity of exponent bits from 8 bits to 11 bits, and the quantity of exponent bits is the same as a quantity of exponent bits of the FP64. For a mantissa of the first-precision intermediate computation result, 29 zeros are padded after an end bit to extend a quantity of mantissa bits from 23 bits to 52 bits, and the quantity of mantissa bits is the same as a quantity

[00246] The foregoing description is merely an embodiment of this application, but is not intended to limit this application. Any modification, equivalent replacement, or improvement made without departing from the principle of this application shall fall within the protection scope of this application.

CLAIMS

1. An arithmetic logic unit in a processor, wherein the arithmetic logic unit comprises a floating-point number decomposition circuit, a second-precision multiplier, and an accumulator;

the floating-point number decomposition circuit is configured to decompose each input to-be-computed first-precision floating-point number into at least two second-precision floating-point numbers, wherein precision of the second-precision floating-point number is lower than precision of the first-precision floating-point number;

the second-precision multiplier is configured to: receive a combination comprising two second-precision floating-point numbers obtained by decomposing different first-precision floating-point numbers, perform a multiplication operation on the second-precision floating-point numbers in each combination, and output an intermediate computation result corresponding to each combination; and

the accumulator is configured to perform an operation to obtain a computation result for the plurality of to-be-computed first-precision floating-point numbers based on the intermediate computation result corresponding to each combination.

2. The arithmetic logic unit according to claim 1, wherein the operation is a summation operation.

3. The arithmetic logic unit according to claim 1 or 2, wherein the arithmetic logic unit further comprises an exponent adjustment circuit;

the floating-point number decomposition circuit is further configured to output, to the exponent adjustment circuit, an exponent bias value corresponding to each second-precision floating-point number;

the second-precision multiplier is further configured to output, to the exponent adjustment circuit, the intermediate computation result corresponding to each combination; and

the exponent adjustment circuit is configured to: adjust, based on the exponent bias value corresponding to the second-precision floating-point number in each input combination, an exponent of the intermediate computation result corresponding to each input combination; and output an adjusted intermediate computation result to the accumulator.

4. The arithmetic logic unit according to claim 3, wherein the exponent adjustment circuit is configured to: add the exponent bias value corresponding to the second-precision floating-point number in each input combination and the exponent of the intermediate computation result corresponding to each input combination, and output the adjusted intermediate computation result to the accumulator.

5. The arithmetic logic unit according to claim 3, wherein the intermediate computation result is a first-precision intermediate computation result, and the computation result is a first-precision computation result.

6. The arithmetic logic unit according to claim 5, wherein the first-precision floating-point number is a single-precision floating-point number, the second-precision floating-point number is a half-precision floating-point number, the first-precision intermediate computation result is a single-precision intermediate computation result, the first-precision computation result is a single-precision computation result, and the second-precision multiplier is a half-precision multiplier; or

the first-precision floating-point number is a double-precision floating-point number, the second-precision floating-point number is a single-precision floating-point number, the first-precision intermediate computation result is a double-precision intermediate computation result, the first-precision computation result is a double-precision computation result, and the second-precision multiplier is a single-precision multiplier.

7. The arithmetic logic unit according to claim 3, wherein the arithmetic logic unit further comprises a format conversion circuit;

the second-precision multiplier is specifically configured to: perform a multiplication operation on the second-precision floating-point numbers in each combination, and output, to the format conversion circuit, a first-precision intermediate computation result corresponding to each combination;

the format conversion circuit is configured to: perform format conversion on each input first-precision intermediate computation result, and output, to the exponent adjustment circuit, a third-precision intermediate computation result corresponding to each combination, wherein precision of the third-precision intermediate computation result is higher than precision of the first-precision intermediate computation result;

the exponent adjustment circuit is configured to: adjust, based on the exponent bias value corresponding to the second-precision floating-point number in each input combination, an exponent of the third-precision intermediate computation result corresponding to each input combination; and output an adjusted third-precision intermediate computation result to the accumulator; and

the accumulator is configured to: perform a summation operation on the adjusted third-precision intermediate computation results corresponding to all the input combinations, and output a third-precision computation result for the plurality of first-precision floating-point numbers.

8. The arithmetic logic unit according to claim 7, wherein the format conversion circuit is

configured to:

perform zero padding processing on an exponent and a mantissa of each input first-precision intermediate computation result, and output, to the exponent adjustment circuit, the third-precision intermediate computation result corresponding to each combination.

9. The arithmetic logic unit according to claim 7 or 8, wherein the first-precision floating-point number is a single-precision floating-point number, the second-precision floating-point number is a half-precision floating-point number, the first-precision intermediate computation result is a single-precision intermediate computation result, the third-precision intermediate computation result is a double-precision intermediate computation result, the third-precision computation result is a double-precision computation result, and the second-precision multiplier is a half-precision multiplier.

10. The arithmetic logic unit according to claim 3, wherein the arithmetic logic unit further comprises a format conversion circuit;

the second-precision multiplier is configured to: perform a multiplication operation on the second-precision floating-point numbers in each combination, and output, to the format conversion circuit, a third-precision intermediate computation result corresponding to each combination;

the format conversion circuit is configured to: perform format conversion on each input third-precision intermediate computation result, and output, to the exponent adjustment circuit, a first-precision intermediate computation result corresponding to each combination;

the exponent adjustment circuit is configured to: adjust, based on the exponent bias value corresponding to the second-precision floating-point number in each input combination, an exponent of the first-precision intermediate computation result corresponding to each input combination; and output an adjusted first-precision intermediate computation result to the accumulator; and

the accumulator is configured to: perform a summation operation on the adjusted first-precision intermediate computation results corresponding to all the input combinations, and output a first-precision computation result for the plurality of first-precision floating-point numbers.

11. The arithmetic logic unit according to claim 10, wherein the first-precision floating-point number is a double-precision floating-point number, the second-precision floating-point number is a half-precision floating-point number, the third-precision intermediate computation result is a single-precision intermediate computation result, the first-precision intermediate computation result is a double-precision intermediate computation result, the first-precision computation result is a double-precision computation result, and the

second-precision multiplier is a half-precision multiplier.

12. The arithmetic logic unit according to any one of claims 1 to 11, wherein the arithmetic logic unit further comprises a computation mode switching circuit;

the computation mode switching circuit is configured to: when the computation mode switching circuit is set to a second-precision floating-point number computation mode, set the floating-point number decomposition circuit and the exponent adjustment circuit to be invalid;

the second-precision multiplier is configured to: receive a plurality of groups of to-be-computed second-precision floating-point numbers that are input from the outside of the arithmetic logic unit, perform a multiplication operation on each group of second-precision floating-point numbers, and input an intermediate computation result corresponding to each group of to-be-computed second-precision floating-point numbers; and

the accumulator is configured to: perform a summation operation on the intermediate computation results corresponding to all the input groups of to-be-computed second-precision floating-point numbers, and output a computation result for the plurality of groups of to-be-computed second-precision floating-point numbers.

13. A floating-point number multiplication computation method, wherein the method comprises:

obtaining a plurality of to-be-computed first-precision floating-point numbers;

decomposing each to-be-computed first-precision floating-point number to obtain at least two second-precision floating-point numbers, wherein precision of the second-precision floating-point number is lower than precision of the first-precision floating-point number;

determining various combinations comprising two second-precision floating-point numbers obtained by decomposing different first-precision floating-point numbers;

inputting the second-precision floating-point numbers in each combination into a second-precision multiplier to obtain an intermediate computation result corresponding to each combination; and

determining a computation result for the plurality of to-be-computed first-precision floating-point numbers based on the intermediate computation result corresponding to each combination.

14. The method according to claim 13, after the decomposing each to-be-computed first-precision floating-point number to obtain at least two second-precision floating-point numbers, further comprising:

determining an exponent bias value corresponding to each second-precision floating-point number; and

the determining a computation result for the plurality of to-be-computed first-precision

floating-point numbers based on the intermediate computation result corresponding to each combination comprises:

adjusting, based on the exponent bias value corresponding to the second-precision floating-point number in each combination, an exponent of the intermediate computation result corresponding to each combination, to obtain an adjusted intermediate computation result; and

performing a summation operation on the adjusted intermediate computation results corresponding to all the combinations, to obtain the computation result for the plurality of first-precision floating-point numbers.

15. The method according to claim 14, wherein the adjusting, based on the exponent bias value corresponding to the second-precision floating-point number in each combination, an exponent of the intermediate computation result corresponding to each combination, to obtain an adjusted intermediate computation result comprises:

adding the exponent of the intermediate computation result corresponding to each combination of second-precision floating-point numbers and the exponent bias value corresponding to the second-precision floating-point number in each combination, to obtain the adjusted intermediate computation result.

16. The method according to claim 13, wherein the intermediate computation result is a first-precision intermediate computation result, and the computation result is a first-precision computation result.

17. A floating-point number multiplication computation apparatus, wherein the apparatus comprises:

an obtaining module, configured to obtain a plurality of to-be-computed first-precision floating-point numbers;

a decomposition module, configured to decompose each to-be-computed first-precision floating-point number to obtain at least two second-precision floating-point numbers, wherein precision of the second-precision floating-point number is lower than precision of the first-precision floating-point number;

a combination module, configured to determine various combinations comprising two second-precision floating-point numbers obtained by decomposing different first-precision floating-point numbers;

an input module, configured to input the second-precision floating-point numbers in each combination into a second-precision multiplier to obtain an intermediate computation result corresponding to each combination; and

a determining module, configured to determine a computation result for the plurality of to-be-computed first-precision floating-point numbers based on the intermediate computation

result corresponding to each combination.

18. The apparatus according to claim 20, wherein the decomposition module is further configured to:

determine an exponent bias value corresponding to each second-precision floating-point number; and

the determining module is configured to:

adjust, based on the exponent bias value corresponding to the second-precision floating-point number in each combination, an exponent of the intermediate computation result corresponding to each combination, to obtain an adjusted intermediate computation result; and

perform a summation operation on the adjusted intermediate computation results corresponding to all the combinations, to obtain the computation result for the plurality of first-precision floating-point numbers.

19. An electronic device, wherein the electronic device comprises a processor and a memory, the memory stores at least one instruction, and the instruction is loaded and executed by the processor to implement an operation performed in the floating-point number multiplication computation method according to any one of claims 13 to 16.

20. A processor, wherein the processor comprises the arithmetic logic unit according to claims 1 to 12.