

COMPUTER PROCESSOR UTILIZING LOGARITHMIC CONVERSION
AND METHOD OF USE THEREOF

Related Inventions

5

The present invention is related to the following inventions which are assigned to the same assignee as the present invention:

- 10 (1) "Neural Network Utilizing a Logarithmic Function and Method of Using Same", having Serial No. 08/176,601, filed on January 3, 1994.
- (2) "Exponentiator Circuit Utilizing Shift Register and Method of Using Same", having Serial No. 08/401,515, filed on March 10, 1995.
- 15 (3) "Accumulator Circuit and Method of Use Thereof", having Serial No. 08/455,927, filed on May 31, 1995.
- (4) "Logarithm/Inverse-Logarithm Converter and Method of Using Same", having Serial No. 08/381,368, filed on January 31, 1995.
- 20 (5) "Logarithm/Inverse-Logarithm Converter Utilizing Second-Order Term and Method of Using Same", having Serial No. 08/382,467, filed on January 31, 1995.

The subject matter of the above-identified related inventions is hereby incorporated by reference into the
25 disclosure of this invention.

Field of the Invention

The present invention relates generally to computers
30 and, in particular, to a computer processor having multiple processing elements.

Background of the Invention

Computer processors are well known and widely used for a variety of purposes. One application of computer processors is digital signal processing (DSP). By definition, digital signal processing is concerned with the representation of signals by sequences of numbers or symbols and the processing of these signals. DSP has a wide variety of applications and its importance is evident in such fields as pattern recognition, radio communications, telecommunications, radar, biomedical engineering, and many others.

At the heart of every DSP system is a computer processor which performs mathematical operations on signals. Generally, signals received by a DSP system are first converted to a digital format used by the computer processor. Then, the computer processor executes a series of mathematical operations on the digitized signal. The purpose of these operations may be to estimate characteristic parameters of the signal or to transform the signal into a form which is in some sense more desirable. Such operations typically implement complicated mathematics and entail intensive numerical processing. Examples of mathematical operations which may be performed in DSP systems include matrix multiplication, matrix-inversion, Fast Fourier Transforms (FFT), auto and cross correlation, Discrete Cosine Transforms (DCT), polynomial equations, and difference equations in general, such as those used to approximate Infinite Impulse Response (IIR) and Finite Impulse Response (FIR) filters.

Computer processors vary considerably in design and function. One aspect of a processor design is its architecture. Generally, the term computer architecture refers to the instruction set and organization of a processor. An instruction set is a group of programmer-

visible instructions used to program the processor. The organization of a processor, on the other hand, refers to its overall structure and composition of computational resources, for example, the bus structure, memory arrangement, and number of processing elements. A processing element, in a broad sense, is a device in a computer processor that performs logical or arithmetical operations on data it receives. For example, a processing element may be as simple as an adder circuit which sums two values, or it may be as complex as a central processing unit (CPU) which performs a wide variety of different operations.

There are many different architectures, ranging from complex-instruction-set-computer (CISC) to reduced-instruction-set-computer (RISC) based architectures. In addition, some architectures have only one processing element, while others include two or more processing elements. Despite differences in architectures, all computer processors have a common goal, which is to provide the highest performance at the lowest cost. However, the performance of a computer processor is highly dependent on the problem to which the processor is applied, and few, if any, low-cost computer processors are capable of performing the mathematical operations listed above at speeds required for some of today's more demanding applications. For example, MPEG data compression of an NTSC television signal can only be performed using expensive super-computers or special purpose hardware.

Many other applications, such as matrix transformations in real-time graphics, require data throughput rates which exceed the capabilities of inexpensive, single processors, such as microprocessors and commercially available DSP chips. Instead, these applications require the use of costly, multiprocessor or multiple-processor computers. Although multiprocessor

computers typically have higher throughput rates, they also include complex instruction sets and are generally difficult to program.

Thus, there is a need for a computer processor ~~that~~ executes mathematical operations at high speeds, while remaining inexpensive. There is also a need for a computer processor which has a simple instruction set and is easily programmed to provide the desired mathematical operations. In addition, there is a need for a computer processor that enjoys a high data throughput rate. Such a processor should also be small in size and consume relatively little power.

Brief Description of the Drawings

The invention is pointed out with particularity in the appended claims. However, other features of the invention will become more apparent and the invention will be best understood by referring to the following detailed description in conjunction with the accompanying drawings in which:

FIG. 1 illustrates a computer processor in accordance with one embodiment of the present invention.

FIG. 2 illustrates a computer processor in accordance with a preferred embodiment of the present invention.

FIG. 3 shows a conceptual diagram of an adder tree.

FIG. 4 shows a conceptual diagram of an accumulation unit.

FIG. 5 illustrates an adder tree used in conjunction with an accumulation unit.

FIG. 6 illustrates an instruction format in accordance with a preferred embodiment of the present invention.

FIG. 7 shows a flow diagram of a method of using the computer processor of FIG. 1 to generate at least one output signal.

FIG. 8 shows a flow diagram of method of using the computer processor of FIG. 1 in which addresses are distributed to the processing elements.

FIG. 9 shows a flow diagram of a method of using the computer processor of FIG. 1 in which output signals are utilized as feedback.

FIG. 10 shows a flow diagram of a method of using the computer processor of FIG. 2 to generate at least one output signal.

FIG. 11 shows a flow diagram of a method of using the computer processor of FIG. 2 in which addresses are distributed to the processing elements.

FIG. 12 shows a flow diagram of a method of using the computer processor of FIG. 2 in which output signals are utilized as feedback.

Detailed Description of a Preferred Embodiment

Generally, the present invention provides a computer processor which receives a plurality of input signals and, in turn, generates at least one output signal. The input and output signals are typically construed as being binary values that represent numeric data. According to one aspect of the present invention, the computer processor includes: a logarithm converter which converts the input signals to corresponding log values; two or more processing elements which receive the log values and generate as output term values; an inverse-logarithm converter for converting the term values to corresponding converted values; an accumulator for summing ones of the converted values to produce at least one output signal; and a control unit which configures the accumulator to perform operations according to an instruction.

An advantage of the present invention is that it provides a computer processor which is capable of

performing sophisticated mathematical operations at high processing speeds. The processor can be used for matrix multiplication, matrix-inversion, Fast Fourier Transforms (FFT), auto and cross correlation, Discrete Cosine Transforms (DCT), polynomial equations, L1 and L2 norm calculations, and difference equations in general, such as those used to approximate Infinite Impulse Response (IIR) and Finite Impulse Response (FIR) filters. Another advantage of the present invention is that it provides a computer processor which has a high data throughput rate. A further advantage of the present invention is that it provides a computer processor which is efficiently implemented in an integrated circuit, and is consequently small in size and consumes relatively little power.

FIG. 1 illustrates a computer processor in accordance with one embodiment of the present invention. The computer processor comprises a logarithm converter 12, and plurality of processing elements, an inverse-logarithm converter 28, a control unit 38, a feedback switch 36, and an accumulator 32. Three of the plurality of processing elements are represented in FIG. 1 by processing elements 16a, 16b, and 16c.

Operation of the computer processor typically proceeds as follows. Initially, the computer processor is configured for run-time execution. This is accomplished when an instruction is received by the control unit 38 over an instruction bus 40. The control unit 38 then decodes the instruction to generate a first control signal 31 and second control signal 33. The first control signal 31 is used to configure the accumulator 32 to perform specific summing operations, while the second control signal 33 is used to configure the feedback switch 36 in either an open or closed position.

Following initial configuration of the accumulator 32 and the feedback switch 36, a plurality of input signals

may be received by the logarithm converter 12 via an input port 10. The logarithm converter 12 performs a logarithmic conversion on the plurality of input signals to generate a corresponding plurality of log values. The log values are then distributed across a bus 14 to the inputs of the plurality of processing elements 16a-16c. In addition to receiving the input signals, the computer processor may also receive a plurality of addresses corresponding to the input signals. The addresses are distributed across an address bus 8 to the plurality of processing elements 16a-16c. The addresses are used by the processing elements to retrieve stored data values corresponding to the inputs values.

Upon receiving the input signals, each processing element performs one or more operations which result in at least one term value being generated as output. The term values generated by the plurality of processing elements are represented by numerals 18a, 18b, and 18c. The inverse-logarithm converter 28 performs an inverse-logarithmic conversion on the term values 18a-18c to generate a plurality of converted values 30. In turn, the accumulator 32 sums selected ones of the converted values 30 to produce one or more output signals which are available on an output port 34.

The logarithm converter 12 can be any means for performing a logarithmic conversion on the input signals, including a computer program executing on a microprocessor. However, in a preferred embodiment of the present invention the logarithm converter 12 approximates a logarithmic function using linear regression. The logarithm converter 12 includes a lookup table (not shown) and an arithmetic means (not shown) for performing conversions as are well known to those skilled in the art. Input signals and log values may be any n-bit representation of numerical values; however, they are preferably represented by 23-bit floating

point numbers having a format which includes one sign bit, a 6-bit exponent, and a 16-bit mantissa.

When a floating point number is received by the logarithm converter 12, the nine most significant bits of the mantissa are used as an address to retrieve a corresponding entry from the previously mentioned lookup table. The retrieved entry is then applied to the arithmetic means to generate the log value output from logarithm converter 12. Further details of the logarithm converter 12 used in a preferred embodiment are given in above-identified Related Invention No. 4.

In another embodiment of the present invention, the logarithm converter 12 approximates a logarithmic function using a second-order regression. Input signals are represented by IEEE standard 32-bit floating point numbers. Using this format, each input signal has one sign bit, an 11-bit exponent, and a 23-bit mantissa. Further details of the logarithm converter used in this embodiment are given in above-identified Related Invention No. 5.

Each of the processing elements 16a-16c generates as output at least one term value 18a-18c, where:

$$y = w + \sum_{i=m}^n g_i x_i \quad \text{Equation 1}$$

25

where y represents the term value, w represents a coefficient value, x_i represents the i th log value received by the processing element, g_i represents an exponent value corresponding to the i th log value, and i , m and n are any integers.

30

In a preferred embodiment, a processing element computes a term value using a digital circuit which

implements shift-based multiplication as described in above-identified Related Invention No. 2.

Each processing element may include a cache memory (not shown) for storing at least one exponent and coefficient value. In this case, each address received across the address bus 8 corresponds to one of the input signals and is used to retrieve from the cache memory a coefficient and exponent value which correspond to the input signal.

The inverse-logarithm converter 28 can be any means for performing an inverse-log conversion on the term values 18a-18c, including a computer program running on a microprocessor or the inverse-logarithm converter as described in above-identified Related Invention No. 5. However, the inverse-logarithm converter 28 preferably has a structure identical to the inverse-logarithm converter disclosed in above-identified Related Invention No. 4, which structure includes a lookup table (not shown) and an arithmetic means (not shown). Optionally, the term values 18a-18c are represented by a binary value having an integer portion and a fractional portion, while the converted values are represented by 23-bit floating point numbers having a format which includes one sign bit, a 6-bit exponent, and a 16-bit mantissa, however, other structures are possible such as a 32-bit floating point format which is IEEE compatible. The fractional portion of the term values 18a-18c is used to retrieve entries from the lookup table. The retrieved entries are then applied to the arithmetic means to generate the corresponding converted values. Further details of the preferred inverse-logarithm converter can be found in above-identified Related Invention No. 4.

The accumulator 32 includes a plurality of summing circuits and configurable data paths, or routing paths. Essentially, the first control signal 31 enables various

data paths for selected converted values to the inputs of the adder circuits. The first control signal 31 also enables data paths which route adder circuit outputs to other adder circuit inputs or to an output port 34.

5 preferred embodiment of the present invention, the accumulator 32 has the structure and function of the accumulator circuit disclosed in above-identified Related Invention No. 3.

The accumulator may be configured to perform any one of four different functions. First, the accumulator 32 may be configured to function as an adder tree. In this configuration, the accumulator 32 generates output signals by summing converted values corresponding to selected ones of the plurality of processing elements. Second, the
15 accumulator 32 may be configured to function as a plurality of individual accumulators. Under this configuration, each of the individual accumulators generates output signals by summing converted values from a designated one of the processing elements. Third, the accumulator 32 may be
20 configured to function as an adder tree used in conjunction with an accumulation unit. In this configuration, the adder tree generates a sequence of adder outputs, over time, by summing converted values corresponding to selected ones of the processing elements, and the accumulation unit
25 then sums the sequence of adder outputs to produce output signals. Under the fourth configuration, the accumulator 32 simply passes the converted values as output signals.

In addition to producing the output signals, the accumulator 32 can also generate output addresses which
30 correspond to the output signals. The output addresses are made available on the output port 34 simultaneously with the corresponding output signals and can be used by external devices to identify individual output signals. In one embodiment of the present invention, the output
35 addresses are stored in read-only-memories (ROMs, not

shown) which are located in the processing elements 16a-16c. The processing elements 16a-16c pass the output addresses to the accumulator across respective buses 19a-19c. Each term value produced by the processing elements has an output address associated with it. In circumstances where converted values from different processing elements are combined into an output signal, the accumulator 32 selects one output address from a plurality of received output addresses and then associates the selected output address with the output signal. For instance, if the accumulator 32 is configured to produce an output signal by summing the converted values from processing elements one and two 16a-16b, then the accumulator could associate the output address from processing element one 16a with the output signal.

The feedback switch 36 provides output signals to the logarithm converter 12 in accordance with the second control signal 33 from the control unit 38. The condition of the second control signal 33 sets the feedback switch in either the open or closed position. In the closed position, output signals are provided as feedback. The condition of the second control signal 33 depends on the instruction decoded by the control unit 38. Instructions can operate the feedback switch in one of three ways. An instruction may either: (1) maintain the feedback switch 36 in the open position, thus providing no feedback; (2) close the feedback switch 36 after a predetermined interval; or (3) periodically open or close the feedback switch 36 at predetermined times during operation of the computer processor.

Feedback of output signals allows the computer processor to operate with much higher throughput during certain operations, such as computing L1 and L2 norms or an Infinite Impulse Response (IIR) digital filter, etc.

Although the computer processor of the present invention is preferably implemented as an integrated circuit, such as an ASIC, it will be understood by one of ordinary skill in the art that the computer processor of the present invention may be implemented in either hardware or software, or any combination thereof. For example, one version of the present invention could implement the logarithm converter 12 as a software program which is executable on a separate processor, such as a microprocessor, while the remaining elements of the computer processor could be included in a custom-designed integrated circuit which is connected to the separate processor.

FIG. 2 illustrates a computer processor in accordance with a preferred embodiment of the present invention. In addition to the elements shown in FIG. 1, the computer processor preferably includes a multiplexer 45 and a de-multiplexer 53. The multiplexer 45 selects, according to a third control signal 46, input signals from either a first input port 42 or a second input port 44. The de-multiplexer 53 transmits, according to a fourth control signal 52, output signals to either a first output port 54 or a second output port 56. In addition to transmitting the output signals, the de-multiplexer 53 can also transmit output addresses to either of the output ports. The third and fourth control signals 46, 52 are generated by the control unit 39 when it decodes an instruction 40.

The dual input and output ports allow the computer processor to receive input signals from two different data streams and provide output signals to either of two receivers. This significantly improves the flexibility of the computer processor in many applications. For instance, two data streams having different data rates and processing requirements can be overlapped by the computer processor,

thus increasing the overall data throughput of the processor.

FIG. 3 shows a conceptual diagram of an adder tree 62. Adder tree 62 comprises a plurality of adder circuits which are connected together in the fashion shown to combine a plurality of adder tree inputs 60 into a single sum on an adder tree output 64. Many different digital adders exist which may be used to implement each adder circuit, including a carry-look-ahead or ripple adder. Each adder circuit sums two operands to produce a sum; thus, an adder tree requires $n-1$ adder circuits for n inputs. In the example shown, the adder tree 62 includes seven adder circuits and can sum eight inputs.

FIG. 4 shows a conceptual diagram of an accumulation unit 70. The accumulation unit comprises an adder circuit 66 for producing a sum and a register 68 for storing the sum. In addition, the register 68 provides the sum as input to the adder circuit 66. This allows the accumulation unit to sum a sequence of inputs. By setting the output of the register 68 to zero, the adder circuit 66 will simply pass inputs to the output of the accumulation unit.

FIG. 5 illustrates the adder tree 62 used in conjunction with the accumulation unit 70. This structure allows the adder tree inputs 60 to be summed, over a period of time, into a single value. A first set of input signals is received on the adder tree inputs 60 and then combined into a single sum on the adder tree output 64. The adder tree output 64 is then summed with the contents of register 68 to produce a further sum. In turn, the further sum is stored in the register 68, and the process is repeated when another set of inputs arrives at the adder tree inputs 60.

FIG. 6 illustrates an instruction format in accordance with a preferred embodiment of the present invention. Although one of ordinary skill will appreciate the fact

that a computer instruction may assume many different formats and retain the same meaning, the instruction format of the present invention is preferably a 19-bit digital word which includes the following fields: an instruction ID, I/O mode, chip ID, offset, and address. The instruction ID is a 5-bit slice which indicates the identity of the instruction. Thus, the preferred format allows the instruction set of the computer processor to comprise up to 32 different instructions. As shown in FIG. 6, the instruction ID includes bits 14-18 of an instruction. The I/O mode is a 2-bit slice which is decoded to generate the third control signal 46 and the fourth control signal 52. These signals, in turn, configure the input and output ports of the computer processor. The I/O mode is located at bits 12-13 of an instruction. Bits 7-11 of an instruction give the chip ID. In a preferred embodiment, a computer processor has a corresponding 5-bit identification number. The computer processor decodes only those instructions with matching chip IDs. The chip ID field allows an instruction to be distributed to a plurality of computer processors on a common bus with only computer processors having matching identification numbers responding to the instruction. Bits 0-6 of an instruction give the instruction offset. The instruction offset is decoded to generate the first control signal 31, which in turn configures the accumulator 32.

FIG. 7 shows a flow diagram of a method of using the computer processor of FIG. 1 to generate at least one output signal. In box 100, a plurality of input signals is received on the input port 10 of the computer processor. Next, in box 102, the logarithm converter 12 performs a logarithmic conversion on the input signals to generate a corresponding plurality of log values. In box 104, the log values are distributed to the plurality of processing elements 16a-16c. In box 106, each of the processing

elements generates at least one term value as a result of receiving the log values. In box 108, an inverse-logarithmic conversion is performed on the term values to generate a plurality of converted values. In box 110, instruction is decoded by the control unit 38 to generate the first control signal 31 which configures the accumulator 32 to perform summing operations. Next, in box 112, the accumulator 32 sums ones of the converted values to produce the at least one output signal on output port 34.

Referring to box 110, the accumulator 32 is configured to perform any of the four different functions as described above for FIG. 1. In one embodiment of the present invention, the step of configuring the accumulator 32, box 110, occurs before the step of receiving input signals, box 100.

Referring now to box 112, the computer processor may, in addition to the output signals, generate output addresses which correspond to the output signals. These output addresses can be used by devices coupled to the computer processor to identify individual output signals.

FIG. 8 shows a flow diagram of a method of using the computer processor of FIG. 1 in which addresses are distributed to the processing elements. In contrast to the method shown in FIG. 7, the method shown in FIG. 8 includes steps for processing input signals and their corresponding addresses. In box 120, a plurality of input signals and their associated addresses are received by the computer processor. Next, in box 122, the logarithm converter 12 performs a logarithmic conversion on the input signals to generate a plurality of log values. In box 124, the log values and their respective addresses are distributed to the plurality of processing elements 16a-16c. In box 126, each processing element uses the addresses to retrieve coefficient and exponent values corresponding to the input

signals. Next, in box 128, each of the processing elements generates at least one term value as a result of receiving the log values and retrieving the coefficient and exponent values. In box 130, an inverse-logarithmic conversion is performed on the term values to generate a plurality of converted values. In box 132, an instruction is decoded by the control unit 38 to generate the first control signal 31 which configures the accumulator 32 to perform summing operations. Next, in box 134, the accumulator 32 sums ones of the converted values to produce the at least one output signal on output port 34.

Referring to box 132, the accumulator 32 is configured to perform any of the four different functions as described above for FIG. 1. In one embodiment of the present invention, the step of configuring the accumulator 32, box 132, occurs before the step of receiving input signals, box 120.

Referring now to box 134, the computer processor may, in addition to the output signals, generate output addresses which correspond to the output signals. These output addresses can be used by devices coupled to the computer processor to identify individual output signals.

FIG. 9 shows a flow diagram of a method of using the computer processor of FIG. 1 in which output signals are utilized as feedback. Feedback of output signals allows the computer processor to operate with much higher throughput during certain operations, such as computing L1 and L2 norms or an Infinite Impulse Response (IIR) digital filter. In box 140, a plurality of input signals are received by the computer processor. Next, in box 142, the logarithm converter 12 performs a logarithmic conversion on the input signals to generate a plurality of log values. In box 144, the log values are distributed to the plurality of processing elements 16a-16c. In box 146, each processing element generates at least one term value as a

result of receiving the log values. In box 148, an inverse-logarithmic conversion is performed on the term values to generate a plurality of converted values. In box 150, an instruction is decoded by the control unit 38 to generate the first control signal 31 which configures the accumulator 32 to perform summing operations. Next, in box 152, the accumulator 32 sums ones of the converted values to produce the at least one output signal on output port 34.

- 10 In decision box 154 a check is made to determine whether the output signals are to be provided as feedback. If so, the method returns to box 142 and the output signals are provided as inputs to the logarithm converter 12 via the feedback switch 36. If not, the method terminates.
- 15 The determination of whether or not to provide feedback is based on the instruction decoded by the control unit 38.

Referring to box 150, the accumulator 32 is configured to perform any of the four different functions as described above for FIG. 1. In one embodiment of the present invention, the step of configuring the accumulator 32, box 20 150, occurs before the step of receiving input signals, box 140.

Referring now to box 152, the computer processor may, in addition to the output signals, generate output 25 addresses which correspond to the output signals. These output addresses can be used by devices coupled to the computer processor to identify individual output signals.

FIG. 10 shows a flow diagram of a method of using the computer processor of FIG. 2 to generate at least one 30 output signal. Referring to box 160, the first, third, and fourth control signals are generated by decoding an instruction. In box 162, the third control signal 46 is applied to the multiplexer 45 to select a plurality of input signals from either the first input port 42 or the 35 second input port 44. Next, in box 164, the logarithm

converter 12 performs a logarithmic conversion on the plurality of input signals to generate a corresponding plurality of log values. In box 166, the log values are distributed to the plurality of processing elements 16c. In box 168, each processing element generates at least one term value. In box 170, an inverse-logarithmic conversion is performed on the term values to generate a plurality of converted values. Next, in box 172, the first control signal 31 is applied to the accumulator 32 to configure the accumulator 32 to perform summing operations. In box 174, the accumulator 32 sums selected ones of the plurality of converted values to produce the at least one output signal. In box 176, the fourth control signal 52 is applied to the de-multiplexer 53 in order to transmit the at least one output signal to either the first output port 54 or second output port 56.

FIG. 11 shows a flow diagram of a method of using the computer processor of FIG. 2 in which addresses are distributed to the processing elements. Referring to box 200, the first, third, and fourth control signals are generated by decoding an instruction. In box 202, the third control signal 46 is applied to the multiplexer 45 to select a plurality of input signals from either the first input port 42 or the second input port 44. Next, in box 204, a plurality of addresses corresponding to the input signals are received. In box 206, the logarithm converter 12 performs a logarithmic conversion on the plurality of input signals to generate a corresponding plurality of log values. In box 208, the log values and their corresponding addresses are distributed to the plurality of processing elements 16a-16c. In box 210, each processing element uses the addresses to retrieve coefficient and exponent values corresponding to the input signals and then generates at least one term value. In box 212, an inverse-logarithmic conversion is performed on the term values to generate a

plurality of converted values. Next, in box 214, the first control signal 31 is applied to the accumulator 32 to configure the accumulator 32 to perform summing operations. In box 216, the accumulator 32 sums selected ones of the plurality of converted values to produce the at least one output signal. In box 218, the fourth control signal 52 is applied to the de-multiplexer 53 in order to transmit the at least one output signal to either the first output port 54 or second output port 56.

FIG. 12 shows a flow diagram of a method of using the computer processor of FIG. 2 in which output signals are utilized as feedback. Referring to box 230, the first, third, and fourth control signals are generated by decoding an instruction. In box 232, the third control signal 46 is applied to the multiplexer 45 to select a plurality of input signals from either the first input port 42 or the second input port 44. Next, in box 234, the logarithm converter 12 performs a logarithmic conversion on the plurality of input signals to generate a corresponding plurality of log values. In box 236, the log values are distributed to the plurality of processing elements 16a-16c. In box 238, each processing element generates at least one term value. In box 240, an inverse-logarithmic conversion is performed on the term values to generate a plurality of converted values. Next, in box 242, the first control signal 31 is applied to the accumulator 32 to configure the accumulator 32 to perform summing operations. In box 244, the accumulator 32 sums selected ones of the plurality of converted values to produce the at least one output signal.

In decision box 246 a check is made to determine whether the output signals are to be provided as feedback. If so, the method returns to box 234 and the output signals are provided as inputs to the logarithm converter 12 via the feedback switch 36. If not, the method terminates.

The determination of whether or not to provide feedback is based on the instruction decoded by the control unit 39.

In box 248, the fourth control signal 52 is applied to the de-multiplexer 53 in order to transmit the at least one output signal to either the first output port 54 or second output port 56.

Thus, in summary, there has been described herein a concept, as well as several embodiments, including a preferred embodiment, of a computer processor which performs logarithmic conversions on input signals prior to performing arithmetic operations on the input signals, thus allowing the processor to perform mathematical operations at dramatically greater speeds while reducing the size and complexity of the processor in an integrated circuit. By operating on logarithmic values rather than numeric input signals, the processing elements can be implemented using significantly less silicon space.

Because various embodiments of the present invention provide a computer processor which includes dual input and output ports, they perform with greatly increased data throughput rates.

While specific embodiments of the present invention have been shown and described, it will be apparent to those skilled in the art that the disclosed invention may be modified in numerous ways and may assume many embodiments other than the preferred form specifically set out and described above.

Accordingly, it is intended by the appended claims to cover all modifications of the invention which fall within the true spirit and scope of the invention.

What is claimed is:



CLAIMS

1. In a computer processor having a first input port, a second input port, and a plurality of processing elements, a method for generating at least one output signal, the method comprising the following steps:
- generating a first control signal, a third control signal, and a fourth control signal by decoding an instruction;
- applying the third control signal to a multiplexer to select a plurality of input signals from either the first input port or the second input port;
- performing a logarithmic conversion on the plurality of input signals to generate a plurality of log values;
- 15 distributing the plurality of log values to the plurality of processing elements;
- each of the plurality of processing elements generating at least one term value;
- performing an inverse-logarithmic conversion on the at least one term value from each of the plurality of processing elements to generate a plurality of converted values;
- 20 applying the first control signal to an accumulator to configure the accumulator to perform summing operations;
- 25 summing ones of the plurality of converted values to produce the at least one output signal; and
- applying the fourth control signal to a de-multiplexer to transmit the at least one output signal to either a first output port or a second output port.



5. The method of claim 1, wherein the step of applying the first control signal includes the substep of: configuring the accumulator to generate an output signal by summing converted values corresponding to selected ones of the plurality of processing elements.

6. The method of claim 1, wherein the step of applying the first control signal includes the substep of: configuring the accumulator to generate a plurality of output signals, wherein each of the plurality of output signals is generated by summing converted values corresponding to one of the plurality of processing elements.

7. The method of claim 1, wherein step of applying the first control signal includes the substep of: configuring the accumulator to generate an output signal by summing converted values corresponding to selected ones of the plurality of processing elements to produce a sequence of adder outputs, over time, and summing the sequence of adder outputs to produce the output signal.

8. The method of claim 1, further comprising the step of: generating at least one output address which corresponds to the at least one output signal.



9. A computer processor receiving a plurality of input signals and generating at least one output signal, which comprises:

5 a logarithm converter for performing a logarithmic conversion on the plurality of input signals to generate a plurality of log values;

10 a plurality of processing elements, each of the plurality of processing elements having an input and generating as output at least one term value y , wherein

$$y = w + \sum_{i=m}^n g_i x_i,$$

15 wherein w represents a coefficient value, g_i represents an exponent value, x_i represents one of the plurality of log values, m and n are integers, and i is an integer in the range m to n ;

20 a bus for distributing the plurality of log values to the inputs of the plurality of processing elements;

an inverse-logarithm converter for performing an inverse-logarithmic conversion on the at least one term value from each of the plurality of processing elements to generate a plurality of converted values;

25 an accumulator for summing ones of the plurality of converted values to produce the at least one output signal; and

30 a control unit, operatively coupled to the accumulator, for configuring the accumulator to perform operations according to an instruction which is decodable by the control unit.



10. A method for using a computer processor, having an input port and a plurality of processing elements, to generate at least one output signal, the method comprising the following steps:

- 5 (a) receiving a plurality of input signals on the input port;
- (b) performing a logarithmic conversion on the plurality of input signals to generate a plurality of log values;
- 10 (c) distributing the plurality of log values to the plurality of processing elements;
- (d) each of the plurality of processing elements generating at least one term value;
- (e) performing an inverse-logarithmic conversion on
15 the at least one term value from each of the plurality of processing elements to generate a plurality of converted values;
- (f) decoding an instruction to configure an accumulator to perform summing operations; and
- 20 (g) summing ones of the plurality of converted values to produce the at least one output signal.