

EXPONENTIATION CIRCUIT UTILIZING SHIFT MEANS
AND METHOD OF USING SAME

Related Inventions

5

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

(1) "Neural Network Utilizing Logarithmic Function
10 and Method of Using Same", having Serial No. 08/176,601, filed on January 3, 1994.

(2) "Logarithm/Inverse-Logarithm Converter Utilizing Second-Order Term and Method of Using Same", having Serial No. 08/382,467, filed on January 31, 1995.

15 (3) "Computer Processor Utilizing Logarithmic Conversion and Method of Use Thereof", having Serial No. 08/403,158, filed on March 13, 1995.

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

Technical Field

The present invention relates generally to electronic
25 circuits and, in particular, to a circuit for generating an exponential signal in response to an input signal.

Background of the Invention

30 It is quickly becoming a common task in digital computers to compute exponential values, x^y . Exponential values are used in a wide variety of computer applications, such as non-linear control, digital signal processing, simulation, and encryption. However, conventional
35 techniques for computing exponential values present several disadvantages.

One approach commonly used for computing exponential values is to provide a binary multiplier circuit as part of the computer. In fact, many of today's microprocessors include one or more multipliers in their core logic.

5 Typically, the multipliers are capable of multiplying two operands at a time. To compute an exponential value x^9 , the input value, x , is presented to both inputs of the multiplier. In turn, the multiplier produces an output of x^2 . If a higher order exponential value is desired, for
10 instance, x^9 , the output of the multiplier is fed back to one or both of the multiplier inputs and the multiplication-feedback cycle is repeated until the desired exponential value is computed.

This approach has several drawbacks. First,
15 conventional binary multiplier circuits require relatively large amounts of space in an integrated circuit. Furthermore, they are typically slow, in some cases requiring a large number of clock cycles to complete a single multiplication, and they usually consume a large
20 amount of power when compared to other arithmetic circuits, such as adders. Another limitation of this approach is that it requires additional multiplication cycles to compute exponential values having a power greater than two. For example, computing an exponential value of x^{16} would
25 require a minimum of four multiplication cycles using a single multiplier circuit. Consequently, such a computation would dramatically decrease the throughput of a computer.

Another known technique for computing exponential
30 values is to provide an analog circuit which determines an exponential value from an analog input value. Typically, the exponential value is also an analog signal which must be converted to a binary word by an analog-to-digital converter before being used by a computer. Although analog
35 exponentiation circuits usually require less time to compute an exponential value than the above-described technique of recursively multiplying binary words, they

are generally expensive and difficult to integrate with a digital computer.

In essence, there is a need for a device to compute an exponential value, wherein the device computes an exponential value in substantially less time, require less space and power on an integrated circuit, and integrates more economically with a digital computer than the conventional techniques described above.

10 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 block diagram of an exponentiation circuit in accordance with one embodiment of the present invention.

FIG. 2 illustrates a block diagram of an exponentiation circuit in accordance with another embodiment of the present invention.

FIG. 3 is a table illustrating examples of exponent codes which can be used with the exponentiation circuit of FIG. 2.

FIG. 4 is a second table illustrating alternative examples of exponent codes which can be used with the exponentiation circuit of FIG. 2.

FIG. 5 illustrates a block diagram of an exponentiation circuit in accordance with a further embodiment of the present invention.

FIG. 6 illustrates a block diagram of an exponentiation circuit in accordance with a preferred embodiment of the present invention.

FIG. 7 is a table illustrating examples of exponent codes which can be used with the exponentiation circuit of FIG. 6.

FIG. 8 illustrates a block diagram of an
5 exponentiation circuit which includes a plurality of compute elements.

FIG. 9 illustrates a block diagram of a computer system which includes an exponentiation circuit that conforms with an embodiment of the present invention.

10 FIG. 10 illustrates a flow diagram of a method of generating an exponential signal from an input signal.

FIG. 11 illustrates a flow diagram of a method of generating a term which includes a plurality of exponential signals generated from a sequence of inputs.

15

Detailed Description of the Preferred Embodiments

It is an advantage of the present invention to provide an exponentiation circuit having a high data throughput
20 rate. Another advantage of the present invention is to provide an exponentiation circuit which requires substantially less power and space on an integrated circuit. A further advantage of the present invention is to provide an exponentiation circuit which is economical to
25 implement and easy to integrate with a microprocessor core or other digital logic on a single chip.

Generally, the present invention provides circuit for computing an exponential signal from an input signal. This is accomplished by using a logarithm converter to generate
30 a binary log value from an input signal. A shift register then shifts the bits of the binary log value to generate an intermediate value. The shifting operation is equivalent to multiplying the binary log value by a factor which is a power of two. The intermediate value is then combined with
35 a stored value, and the exponential signal is generated by performing an inverse-logarithmic conversion on the combined value.

FIG. 1 illustrates a block diagram of an exponentiation circuit 40 in accordance with one embodiment of the present invention. The exponentiation circuit 40 comprises a log circuit 4, a compute element 9, and an inverse-log circuit 34. The compute element 9 includes a shift register 8, a P/I unit 12, an adder circuit 24, a register 28, and a control unit 16. The log circuit 4 receives input signals on its input 2 and converts each input signal to a binary log value 6 which has a plurality of bits. Upon receiving a binary log value 6, the shift register 8 performs an arithmetic shift on the input to generate an intermediate value 10. The distance of the shift is determined by a shift signal 18 which is produced by the control unit 16. After receiving the intermediate value 10, the P/I unit 12 either passes the intermediate value 10 to bus 14 or inverts the intermediate value 10 and then passes the inverted value to bus 14. The function of the P/I unit 12 is determined by a control signal 20 which is provided by the control unit 16. The adder circuit 24 sums a stored value, received across feedback path 30, the value provided on bus 14, and a carry bit 22 to produce a combined value on output 26. The combined value is stored in the register 28. The register 28 provides the combined value on a bus 32 to the inverse-log circuit 34 and as the stored value on the feedback path 30. The inverse-log circuit 34 produces the exponential signal on output 36 by taking the inverse-logarithm of the combined value.

The log circuit 4 may be any means which converts an input signal to a binary value representing the logarithm of the input signal. For example, the log circuit may be an analog circuit which includes an amplifier having a logarithmic transfer function and an analog-to-digital converter which converts the output of the amplifier to a binary word. In such an example, the input signal could be a current or voltage which varies continuously over the dynamic range of the amplifier. However, in a preferred embodiment, the log circuit 4 includes a log converter as

disclosed in above-identified Related Invention No. 2. This log converter receives input signals represented by IEEE standard 32-bit floating point values and implements a second-order polynomial to generate binary log values
5 having an 8-bit integer portion and a 23-bit fractional portion. The sign bit of each input signal is passed (not shown) directly to the inverse-log circuit 34 where it is then recombined with the corresponding exponential signal, which is also represented by an IEEE standard 32-bit
10 floating point value.

The inverse-log circuit 34 may be any means which receives as input a binary word and produces as output the inverse-logarithm of the binary word. However, in a preferred embodiment, the inverse-log circuit 34 includes
15 an inverse-logarithm converter as disclosed in above identified Related-Invention No. 2. This inverse-log converter receives both the fractional and integer portions of the combined values and, in turn, generates exponential signals that are represented by IEEE 32-bit floating point
20 values. Such an inverse-log converter includes a means (not shown) for recombining a floating point exponential signal with its respective sign bit.

In a preferred embodiment, the log circuit 4, compute element 9, and inverse-log circuit 34 are implemented on a
25 single integrated circuit. However, one of ordinary skill in the art will realize that either the log circuit 4 or the inverse-log circuit 34 may be implemented in many other ways and still conform to the teachings of the present invention. For instance, either circuit may be implemented
30 as software running on a general purpose processor, such as a microprocessor, or either may be implemented as a stand-alone analog circuit. Furthermore, one of ordinary skill will appreciate that the logarithmic and inverse-
logarithmic conversions may be based on any number system.

35 The shift register 8 is used to shift the values it receives in a bit-wise fashion. The shifting operation is equivalent to either multiplying or dividing the binary log

value 6 by a factor which is a power of two. The shifted result is the intermediate value 10. Although an embodiment of the present invention does not place any limits on the bit distance which a value may be shifted, a preferred embodiment provides that the shift register 8 can shift a binary log value a bit distance in the range of 0 to 3 bits. The distance that each binary log value 6 is shifted is determined by shift signal 18 which is provided by control unit 16. By shifting the bits of the binary log value 6 toward the most significant bit (MSB) of the value (typically to the left), the shift register 8 produces the intermediate value 10 which is multiplied by a factor that is a power of two. On the other hand, by shifting the bits of the binary log value 6 toward the least significant bit (LSB) of the value (typically to the right), the shift register 8 produces the intermediate value 10 which is divided by a factor that is a power of two. Thus, when the shift register 8 shifts a value to the left, the exponentiation circuit 40 produces an exponential signal having an integer exponent, such as x^2 , x^4 , etc., where x represents the input signal; while shifting to the right produces an exponential signal having a fractional exponent, such as $x^{1/2}$, $x^{1/4}$, etc., where x represents the input signal

The P/I unit 12 inverts the intermediate value 10 in response to the control signal 20 from the control unit 16. By causing the P/I unit 12 to invert the intermediate value 10 and by providing a carry bit 22 corresponding to the inverted value, the control unit 16 converts the intermediate value 10 to its two's complement form, thus allowing the adder circuit 24 to subtract the converted intermediate value, available on bus 14, from the stored value.

The adder circuit 24 and the register 28 function together to allow a sequence intermediate values to be accumulated. In addition, the register 28 can be pre-loaded with a value, such as coefficient value. The adder

circuit 24 sums the values it receives on the bus 14 and the feedback path 30 to generate combined values which are in turn stored in the register 28. A combined value is simply the sum of the values provided on the two inputs of the adder circuit 24. The stored value provided on the feedback path 30 is the contents of the register 28.

The control unit 16 typically stores a plurality of exponent codes (not shown), wherein each exponent code corresponds to an input signal. As each input signal is received by the exponentiation circuit 40, the control unit 16 decodes a corresponding exponent code to generate the shift signal 18, the control signal 20, and the carry bit 22, all of which correspond to the input signal. However, in another version of the present invention, the exponent codes are provided by a means of generating the codes which is connected to the control unit 16 and yet separate from the exponentiation circuit 40, such as a separate memory for storing the exponent codes or a host processor, e.g. a microprocessor connected to the circuit.

In operation, the exponentiation circuit 40 can compute an exponent signal for a single input signal or a term which includes a plurality of exponent signals computed from a sequence of input signals. In either case, computing an exponent signal from a single input signal may require one or more addition cycles. This is accomplished by providing an additional storage means (not shown) for storing the binary log value 6 while the addition cycles execute and providing the stored binary log value to the P/I unit 12 during each cycle. Such a storage means may be incorporated in either the log circuit 4 or in the shift register 8.

The operation of the exponentiation circuit 40 may be illustrated by considering an example of computing an exponential signal of x^7 , where x represents an input signal. The computation of x^7 requires two addition cycles. During each cycle, the shift register 8 provides a different intermediate value 10.

During the first addition cycle, the input signal, x , is converted to a binary log value 6 by the log circuit 4 and provided to the additional storage means and the shift register 8. The control unit generates the shift signal 18 so that the shift register 8 shifts the binary log value 6 a distance of three bits, thus producing an intermediate value 10 which is a multiple of eight of the binary log value 6. The control unit 16 also generates the control signal 20 so that the intermediate value 10 is passed unaffected through the P/I unit 12 to the adder circuit 24 via bus 14. The adder circuit 24 then sums the intermediate value 10 with the value stored in the register 28, which value is provided by feedback path 30. Initially, the stored value may be set to zero or a coefficient value. The contents of the register 28 are then updated to the combined value which is provided on the output 26 of the adder circuit 24.

During the second addition cycle, the control unit 16 generates the carry bit 22 and the shift signal 18 so that the shift register 8 passes the stored binary log value without shifting it. The control unit 16 also generates the control signal 20 so that the P/I unit 12 inverts the intermediate value 10. As a result of the inversion, the adder circuit 24, in effect, subtracts the intermediate value 10 from the stored value provided on the feedback path 30 to produce a final combined value which is a multiple of seven of the binary log value 6. The final combined value is then stored in the register 28 and provided to the inverse-log circuit 34 on bus 32. The inverse-log circuit 34 produces the exponential signal, x^7 , by performing an inverse-logarithmic conversion on the stored value.

FIG. 2 illustrates a block diagram of an exponentiation circuit 100 in accordance with another embodiment of the present invention. The exponentiation circuit 100 performs the same function as the exponentiation circuit 40 shown in FIG. 1 in that it

computes a term which includes one or more exponential signals. However, the exponentiation circuit 100 includes an additional shift register, and thus requires fewer addition cycles than the exponentiation circuit 40 of FIG. 1. The exponentiation circuit 100 comprises a log circuit 4, a compute element 99, and an inverse-log circuit 34. The compute element 99 includes a P/I unit 72, a P/Z unit 60, a first shift register 56, a second shift register 68, a first multiplexer 64, a second multiplexer 76, an adder circuit 80, a de-multiplexer 84, a first register 94, a second register 90, and a control unit 104.

The log circuit 4 receives one or more input signals on its input 2 and converts each input signal to a binary log value which has a plurality of bits. The binary log value is then distributed by bus 54 to the first shift register 56 and the second shift register 68. Upon receiving a binary log value, the first shift register 56 performs an arithmetic shift on the value to generate a first intermediate value on bus 58. The distance of the shift is determined by a first shift signal 106 which is produced by the control unit 104. The second shift register 68 performs an arithmetic shift on the binary log value to generate a second intermediate value on bus 70. The distance of the shift is determined by a second shift signal 108 which is produced by the control unit 104. Both the first shift register 56 and the second shift register 68 may shift binary log values either to the right or left.

After receiving an input on bus 70, the P/I unit 72 either passes the second intermediate value to bus 74 or inverts the intermediate value and then passes the inverted value to bus 74. The function of the P/I unit 72 is determined by a second control signal 120 which is provided by the control unit 104. The P/Z unit 60 either passes the first intermediate value or zero to bus 62, depending on the value of a first control signal 110 which is provided by the control unit 104.

- A first multiplexer 64 couples to its output 66 to either the bus 62, a feedback bus 96, or the output 92 of the second register 90. The output 66 of the first multiplexer 64 is determined by a first mux signal 112
- 5 provided by the control unit 104. A second multiplexer 76 couples to its output 78 to either the bus 74, or the output 92 of the second register 90. The output 78 of the second multiplexer 76 is determined by a second mux signal 116 provided by the control unit 104.
- 10 The adder circuit 80 sums the outputs 66 and 78 of the first and second multiplexers 64, 76. The de-multiplexer 84 provides the output 82 of the adder circuit 80 to either the second register 90 or the first register 94, depending on the value of the de-mux signal 118. The first register
- 15 94 provides its contents, which is a sum, on feedback bus 96 and output 98, while the second register provides its contents as a combined value on its output 92. The inverse-log circuit 34 generates a term, which represents one or more exponential signals, on output 36 by taking the
- 20 inverse-logarithm of the sum.
- The control unit 104 typically stores a plurality of exponent codes (not shown), where each exponent code corresponds to an input signal. As each input signal is received by the exponentiation circuit 100, the control
- 25 unit 104 decodes a corresponding code to generate the first and second control signals 110, 120; the first and second shift signals 106, 108; the first and second mux signals 112 and 116, de-mux signal 118, and a carry bit 114. However, in other versions of the present invention, the
- 30 exponent codes are provided by a means of generating the codes which is connected to the control unit 104 and yet separate from the exponentiation circuit 100, such as a separate memory for storing the exponent codes or a host processor, e.g. a microprocessor connected to the circuit.
- 35 In operation, the exponentiation circuit 100 can compute an exponent value for a single input signal or a term which includes a plurality of exponent signals

computed from a sequence of input signals. In either case, computing an exponent signal from a single input signal requires only one addition cycle in a preferred embodiment. However, computing a term which includes a plurality of
5 exponent value requires two addition cycles for every input signal. This is explained as follows. When an input signal is received, the result of the first addition cycle is stored in the second register 90 as the combined value. This is accomplished by the control unit 104 asserting the
10 first and second mux signals 112, 116 so that the outputs of the P/I unit 72 and the P/Z unit 60 are provided to the adder circuit 80. In addition, the control unit 104 asserts the de-mux signal 118 to select the second register 90.

15 During the second cycle, a stored value and the combined value are provided to the inputs of the adder circuit 80 via the feedback bus 96 and the output 92, respectively. The sum produced by the second addition cycle then updates the stored value in the first register
20 94. The second cycle is accomplished by the control unit 104 asserting the first and second mux signals 112, 116 so that the output 92 and the feedback bus 96 are provided as inputs to the adder circuit 80. The control unit 104 then asserts the de-mux signal 118 to select the first register
25 94 so that the result of the cycle is placed in the first register 94.

In operation, the stored value in the first register 94 is typically the sum of the intermediate values generated from preceding input signals in a sequence.
30 Initially, however, the stored value may be a coefficient value which is the logarithm of a constant. The first and second registers 94, 90 may also be preset to zero.

FIG. 3 is a table illustrating examples of exponent codes which can be used with the exponentiation circuit 100
35 of FIG. 2. The table lists the codes and their corresponding control and shift signals which are used in a preferred embodiment of the present invention. The table

also illustrates the operations of the adder circuit 80 and the value of the carry bit 114 in response to the each of the exponent codes. Each row in the table gives the values of the signals and the carry bit corresponding to one of the exponent codes.

For example, in the first row, the exponent code "000" indicates that x^{-1} is to be generated by exponentiation circuit 100. If the exponent code for an input signal is "000", then the second intermediate value is equated to the binary log value, the first intermediate value is set to zero, and the combined value is produced by subtracting the second intermediate value from the first intermediate value.

In the second row, the exponent code "001" indicates that x^1 is to be generated. If the exponent code is "001" for an input signal, then the second intermediate value is equated to the binary log value, the first intermediate value is set to zero, and the combined value is produced by adding the second intermediate value to the first intermediate value.

In the third row, the exponent code "010" indicates that x^2 is to be generated. If the exponent code is "010" for an input signal, then the exponent code indicates that x^2 is to be generated, then the second intermediate value is generated by shifting the binary log value a distance of one bit to the left, the first intermediate value is set to zero, and the combined value is produced by adding the second intermediate value to the first intermediate value.

In the fourth row, the exponent code "011" indicates that x^3 is to be generated. If the exponent code is "011" for an input signal, then the second intermediate value is generated by shifting the binary log value a distance of one bit to the left, the first intermediate value equated to the binary log value, and the combined value is produced by adding the second intermediate value to the first intermediate value.

In the fifth row, the exponent code "100" indicates that x^4 is to be generated. If the exponent code is "100" for an input signal, then the second intermediate value is generated by shifting the binary log value a distance of two bits to the left, the first intermediate value is set to zero, and the combined value is produced by adding the second intermediate value to the first intermediate value.

In the sixth row, the exponent code "101" indicates that x^5 is to be generated. If the exponent code for an input signal is "101", then the second intermediate value is generated by shifting the binary log value a distance of two bits to the left, the first intermediate value is equated to the binary log value, and the combined value is produced by adding the second intermediate value to the first intermediate value.

In the seventh row, the exponent code "110" indicates that x^6 is to be generated. If the exponent code for an input is "110", then the second intermediate value is generated by shifting the binary log value a distance of two bits to the left, the first intermediate value is generated by shifting the binary log value a distance of one bit to the left, and the combined value is produced by adding the second intermediate value to the first intermediate value.

In the eighth row, the exponent code "111" indicates that x^7 is to be generated. If the exponent code for an input signal is "111", then the second intermediate value is equated to the binary log value, the first intermediate value is generated by shifting the binary log value a distance of three bits to the left, and the combined value is produced by subtracting the second intermediate value from the first intermediate value.

FIG. 4 is a first table illustrating alternative examples of exponent codes which can be used with the exponentiation circuit of FIG. 2. The table lists the codes and their corresponding control and shift signals. The table also illustrates the operations of the adder

circuit 80 and the value of the carry bit 114 in response to the each of the exponent codes. Each row in the table gives the values of the signals and the carry bit corresponding to one of the exponent codes.

5 For example, in the first row, the exponent code "000" indicates that x^{-1} is to be generated by exponentiation circuit 100. If the exponent code for an input signal is "000", then the second intermediate value is equated to the binary log value, the first intermediate value is set to
10 zero, and the combined value is produced by subtracting the second intermediate value from the first intermediate value.

 In the second row, the exponent code "001" indicates that $x^{-1/2}$ is to be generated. If the exponent code is
15 "001" for an input signal, then the second intermediate value is generated by shifting the binary log value one bit to the right, the first intermediate value is set to the binary log value, and the combined value is produced by subtracting the first intermediate value from the second
20 intermediate value.

 In the third row, the exponent code "010" indicates that $x^{1/2}$ is to be generated. If the exponent code is
 "010" for an input signal, then the second intermediate value is shifted one bit to the right, the first
25 intermediate value is set to zero, and the combined value is produced by adding the second intermediate value to the first intermediate value.

 In the fourth row, the exponent code "011" indicates that x^1 is to be generated. If the exponent code is "011"
30 for an input signal, then the second intermediate value is equated to the binary log value, the first intermediate value is set to zero, and the combined value is produced by adding the second intermediate value to the first intermediate value.

35 In the fifth row, the exponent code "100" indicates that x^2 is to be generated. If the exponent code is "100" for an input signal, then the exponent code indicates that

x^2 is to be generated, then the second intermediate value is generated by shifting the binary log value a distance of one bit to the left, the first intermediate value is set to zero, and the combined value is produced by adding the
5 second intermediate value to the first intermediate value.

In the sixth row, the exponent code "101" indicates that x^3 is to be generated. If the exponent code is "101" for an input signal, then the second intermediate value is generated by shifting the binary log value a distance of
10 one bit to the left, the first intermediate value equated to the binary log value, and the combined value is produced by adding the second intermediate value to the first intermediate value.

In the seventh row, the exponent code "110" indicates
15 that x^4 is to be generated. If the exponent code is "110" for an input signal, then the second intermediate value is generated by shifting the binary log value a distance of two bits to the left, the first intermediate value is set to zero, and the combined value is produced by adding the
20 second intermediate value to the first intermediate value.

In the eighth row, the exponent code "111" indicates that x^5 is to be generated. If the exponent code for an input signal is "111", then the second intermediate value is generated by shifting the binary log value a distance of
25 two bits to the left, the first intermediate value is equated to the binary log value, and the combined value is produced by adding the second intermediate value to the first intermediate value.

FIG. 5 illustrates a block diagram of an
30 exponentiation circuit 192 in accordance with a further embodiment of the present invention. The exponentiation circuit 192 performs the same function as the exponentiation circuit 100 shown in FIG. 2 in that it computes a term which includes one or more exponential
35 signals. However, exponentiation circuit 192 includes an second adder circuit 166, and thus requires only one addition cycle per input signal. Although the

exponentiation circuit 192 of FIG. 5 may be faster than the exponentiation circuit 100 of FIG. 2, it consumes more space when implemented as an integrated circuit.

The exponentiation circuit 192 comprises a log circuit 4, a compute element 191, and an inverse-log circuit 34.
5 The compute element 191 includes a P/I unit 160, a P/Z unit 150, a first shift register 146, a second shift register 156, a first adder circuit 154, a second adder circuit 166, a register 170, and a control unit 180.

10 The log circuit 4 receives one or more input signals on its input 2 and converts each input signal to a binary log value which has a plurality of bits. The binary log value is then distributed by bus 144 to the first shift register 146 and the second shift register 156. Upon
15 receiving a binary log value, the first shift register 146 performs an arithmetic shift on the input to generate a first intermediate value on bus 148. The distance of the shift is determined by a first shift signal 182 which is produced by the control unit 180. The second shift
20 register 156 performs an arithmetic shift on the input it receives on bus 144, whereby generating a second intermediate value on bus 158. The distance of the shift is determined by a second shift signal 189 which is produced by the control unit 180.

25 After receiving an input on bus 158, the P/I unit either passes the binary log value to a bus 162 or inverts the binary log value and then passes the inverted value to the bus 162. The function of the P/I unit 160 is determined by a second control signal 188 which is provided
30 by the control unit 180. The P/Z unit 150 either passes the binary log value or zero to bus 152, depending on the value of a first control signal 184 which is provided by the control unit 180.

The first adder circuit 154 sums the outputs of the
35 P/Z unit 150 and the P/I unit 160 which are provided on the buses 152, 162. The output 164 of the first adder circuit 154 is fed to the second adder circuit 166. The second

adder circuit 166 also receives the output of the register 170 on feedback bus 172 and generates a resultant sum which is provided to register 170 on output 168. The inverse-log circuit 34 generates a term representing one or more
5 exponential signals on output 36 by performing an inverse-logarithmic conversion on the sum provided by the register 170 on output 174.

The control unit 180 typically stores a plurality of exponent codes (not shown), where each exponent code
10 corresponds to an input signal. As each input signal is received by the exponentiation circuit 192, the control unit 180 decodes a corresponding code to generate the first and second control signals 184, 188; the first and second shift signals 182, 189; and carry bit 186.

15 In operation, the exponentiation circuit 192 can compute an exponent signal for a single input signal or a term which includes a plurality of exponent signals computed from a sequence of input signals. In either case, computing an exponent signal from a single input signal
20 requires only one addition cycle.

FIG. 6 illustrates a block diagram of an exponentiation circuit in accordance with a preferred embodiment of the present invention. The preferred exponentiation circuit 218 performs the same function as
25 the exponentiation circuits 100, 192 shown in FIGS. 2 and 5 in that it computes a term which includes one or more exponential signals. Moreover, like the exponentiation circuit 192 shown in FIG. 5, the exponentiation circuit 218 of the preferred embodiment requires only one addition
30 cycle per input signal. However, in contrast to the exponentiation circuit 192 of FIG. 5, the preferred exponentiation circuit 218 uses multiplexers instead of shift registers and thus requires less power and silicon space than the exponentiation circuit 192 of FIG. 5.

35 The exponentiation circuit 218 comprises a log circuit 4, a compute element 217, and an inverse-log circuit 34. The compute element 217 includes a first multiplexer 199, a

second multiplexer 203, a third multiplexer 210, a first adder circuit 208, a second adder circuit 212, a register 214, and a control unit 201.

5 The log circuit 4 receives one or more input signals on its input 2 and converts each input signal to a binary log value which has a plurality of bits. The binary log value is then distributed by a bus 190 to the first multiplexer 199 and the second multiplexer 203. Each multiplexer selects from one of four inputs. Both the
10 first and second multiplexers are provided a pass input 193a-b and a zero input 194a-b. In addition, the first multiplexer 199 receives a 1-bit shift left input 195 and a 1-bit shift right input 196; while the second multiplexer 203 receives an inverted input 197 and a 2-bit shift left
15 input 198. The inputs to the first and second multiplexers 199, 203 are respectively selected according to a first mux signal 200 and a second mux signal 202.

The various inputs to the first and second multiplexers 199, 203 comprise different bit slices of the
20 binary log value. For instance, the pass input 193a-b is simply the binary log value, whereas the zero input 194a-b is a binary word with all of its bits set to zero. On the other hand, the 1-bit shift left input 195 effectively provides the binary log value having been shifted one bit
25 to the left. This is accomplished by connecting the right-most $n-1$ bits of the bus 192 to the left-most $n-1$ bits of the multiplexer input, where n is the number of bits in the binary log value. Zero is provided to the least significant bit (right-most bit) of the multiplexer input.
30 The 2-bit shift left input 198 effectively provides the binary log value having been shifted two bits to the left. This is accomplished by connecting the right-most $n-2$ bits of the bus 192 to the left-most $n-2$ bits of the multiplexer input. Zero is provided to the two least significant bits
35 of the multiplexer input. The 1-bit shift right input 196 effectively provides the binary log value having been shifted one bit to the right. This is accomplished by

connecting the left-most $n-1$ bits of the bus 192 to the right-most $n-1$ bits of the multiplexer input. Zero is provided to the most significant bit (left-most bit) of the multiplexer input. The inverted input 197 is created by
5 inverting the bits of the binary log value. This provides the 1's-complement form of the binary log value.

The first adder circuit 208 sums the outputs of the first and second multiplexers 199, 203. The first adder 154 is also responsive to a carry bit 206 and an adder
10 enable signal 205. The output of the first adder 208 is fed to the third multiplexer 210. The third multiplexer 210 selects either the output of the first multiplexer 199, second multiplexer 203, or first adder circuit 208 according to a third mux signal 204 which is generated by
15 the control unit 201.

The second adder 212 sums the outputs of the third multiplexer 210 and a stored value which is provided on a feedback path 216. The output of the second adder circuit 212 is stored in register 214. Initially, the register 214
20 can be pre-loaded with a value, such as a coefficient value or zero. The output of the register 214 is provided to the feedback path 216 and the inverse-log converter 34. The inverse-log circuit 34 generates a term representing one or more exponential signals on output 36 by performing an
25 inverse-logarithmic conversion on the sum provided by the register 214.

The control unit 201 typically stores a plurality of exponent codes (not shown), where each exponent code corresponds to an input signal. As each input signal is
30 received by the exponentiation circuit 218, the control unit 201 decodes a corresponding code to generate the first, second, and third mux signals 200, 202, and 204; the adder enable signal 205; and the carry bit 206. However, in another version of the present invention, the exponent
35 codes are provided by a means of generating the codes which is connected to the control unit 201 and yet separate from the exponentiation circuit 218, such as a separate memory

for storing the exponent codes or a host processor, e.g. a microprocessor connected to the circuit 218.

In operation, the exponentiation circuit 218 can compute an exponent signal for a single input signal or a term which includes a plurality of exponent signals computed from a sequence of input signals. In either case, computing an exponent signal from a single input signal requires only one addition cycle.

In one embodiment of the present invention, the exponentiation circuits 40, 100, 192, or 218 may be used to implement a term of which the general form is represented by Equation 1 as follows:

$$y = w * x_1^{g_1} * x_2^{g_2} \dots x_n^{g_n} \text{ Equation (1)}$$

15

where x_i represent the input signals received on input 2 and can be a function such as $x_i = f_i(z_j)$, where z_j is any arbitrary variable, and where the indices i and j may be any integers; where y represents the output 36 of the exponentiation circuit; where w represents the coefficient value; where $g_1, \dots, g_n$ represent the exponents for the term; and n is the number of inputs.

FIG. 7 is a table illustrating examples of exponent codes which can be used with the exponentiation circuit of FIG. 6. The table lists the codes and their corresponding mux signals which are used in the preferred embodiment of the present invention. The table also illustrates the values of the carry bit 206 and adder enable signal 205 generated in response to the each of the exponent codes. Each row in the table gives the values of the signals and the carry bit corresponding to one of the exponent codes.

For example, in the first row, the exponent code "000" indicates that x^{-1} is to be generated by exponentiation circuit 218. If the exponent code for an input signal is "000", then the first mux signal 200 is a don't care, the second mux signal 202 is set to select the inverted input

197, the adder enable signal is set to zero, thus disabling the first adder 208, the third mux signal 204 is set to select the output of the second multiplexer 203, and the carry bit 206 is set to one. In effect, the 2's-complement
5 form of the binary log value is generated by the second adder circuit 212.

In the second row, the exponent code "001" indicates that $x^{-1/2}$ is to be generated. If the exponent code is "001" for an input signal, then the first mux signal 200 is
10 set to select the 1-bit shift right input 196, the second mux signal 202 is set to select the inverted input 197, the adder enable signal 205 is set to one, thus enabling the first adder 208, the third mux signal 204 is set to select the output of the first adder 208, and the carry bit 206 is
15 set to one for the first adder 208 and zero for the second adder 212.

In the third row, the exponent code "010" indicates that $x^{1/2}$ is to be generated. If the exponent code is "010" for an input signal, then the first mux signal 200 is
20 set to select the 1-bit shift right input 196, the second mux signal 202 is a don't care, the adder enable signal 205 is set to zero, the third mux signal 204 is set to select the output of the first multiplexer 199, and the carry bit 206 is set to zero.

In the fourth row, the exponent code "011" indicates that x^1 is to be generated. If the exponent code is "011" for an input signal, then the first mux signal 200 is set to select the pass input 193a, the second mux signal 202 is a don't care, the adder enable signal 205 is set to zero,
30 the third mux signal 204 is set to select the output of the first multiplexer 199, and the carry bit 206 is set to zero.

In the fifth row, the exponent code "100" indicates that x^2 is to be generated. If the exponent code is "100" for an input signal, then the first mux signal 200 is set to select the 1-bit shift left input 195, the second mux
35 signal 202 is a don't care, the adder enable signal 205 is

set to zero, the third mux signal 204 is set to select the output of the first multiplexer 199, and the carry bit 206 is set to zero.

5 In the sixth row, the exponent code "101" indicates that x^3 is to be generated. If the exponent code is "101" for an input signal, then the first mux signal 200 is set to select the 1-bit shift left input 195, the second mux signal 202 is set to select the pass input 193b, the adder enable signal 205 is set to one, thus enabling the first
10 adder 208, the third mux signal 204 is set to select the output of the first adder 208, and the carry bit 206 is set to zero.

In the seventh row, the exponent code "110" indicates that x^4 is to be generated. If the exponent code is "110"
15 for an input signal, then the first mux signal 200 is a don't care, the second mux signal 202 is set to select the 2-bit shift left input 198, the adder enable signal 205 is set to zero, the third mux signal 204 is set to select the output of the second multiplexer 203, and the carry bit 206
20 is set to zero.

In the eighth row, the exponent code "111" indicates that x^5 is to be generated. If the exponent code for an input signal is "111", then the first mux signal 200 is set to select the pass input 193a, the second mux signal 202 is
25 set to select the 2-bit shift left input 198, the adder enable signal 205 is set to one, the third mux signal 204 is set to select the output of the first adder 208, and the carry bit 206 is set to zero.

An exponent code may also be provided to indicate that
30 x^0 is to be generated. Although such a code is not shown in FIG. 7, it could be easily implemented by providing an additional exponent code, such as "1000", which when decoded causes the first mux signal 200 to select the zero input 194b, the adder enable signal 205 to be zero, the
35 third mux signal to select the output of the first multiplexer 199, and the carry bit to be zero. The second mux signal 202 is a don't care.

FIG. 8 illustrates a block diagram of an exponentiation circuit which includes a plurality of compute elements. Like the other embodiments to the present invention, the exponentiation circuit of FIG. 8 computes a term which includes one or more exponential signals. This embodiment of the exponentiation circuit includes a log converter 220, a plurality of compute elements 223, and an inverse-log converter 226. The log converter 220 receives one or more input signals on its input and converts each input signal to a binary log value which has a plurality of bits. The binary log value is then distributed by a bus 222 to each of the plurality of compute elements. The outputs of the compute elements are then passed across a bus 224 to the inverse-log converter 226. The inverse-log converter 226 generates a term representing one or more exponential signals on output 36 by performing an inverse-logarithmic conversion on the outputs generated by the plurality of compute elements 223.

The plurality of compute elements may include any of the compute elements 9, 99, 191, 217 depicted in FIGS. 1, 2, 5, and 6.

The exponentiation circuit of the present invention and its various embodiments, which are shown in FIGS. 1, 2, 5, 6, and 8 are preferably implemented with integrated circuits. However, one of ordinary skill in the art will recognize that a programmable logic array or other digital logic device, such as a microprocessor executing software, could also be used to implement the functions performed by the various embodiments of the exponentiation circuit of the present invention.

FIG. 9 illustrates a block diagram of a computer system which includes an exponentiation circuit which conforms with an embodiment of the present invention. The computer system 230 comprises a memory 232, a processor 236, and an exponentiation circuit 239. The memory 232 stores programs and data which are used by the processor 236. In operation, the processor 236 fetches program

instruction from the memory 232 using bus 234. As a result of executing the program instructions, the processor 236 transmits input signals to the exponentiation circuit 239 and subsequently retrieves at least one exponent value from the output of the exponentiation circuit 239. All communication between the processor 236 and the exponentiation 239 take place over a bus 238. The exponentiation circuit 239 may include any of the exponentiation circuits 40, 100, 192, or 218 depicted in FIGS. 1, 2, 5, and 6, respectively.

FIG. 10 illustrates a flow diagram of a method of generating an exponential signal from an input signal. Such a method could be used in an exponentiation circuit having an input signal x and generating an exponential signal x^g .

In box 240, a binary log value having a plurality of bits is generated by performing a logarithmic conversion on the input signal.

In box 242, a plurality of bits of the binary log value are shifted to generate an intermediate value. The shifting operation is equivalent to multiplying the binary log value by a factor which is a power of two.

Next, in box 244, the intermediate value is arithmetically combined with a stored value to produce a combined value. The two values may be combined by either adding the intermediate value to the stored value, or subtracting the intermediate value from the stored value.

In box 246, the stored value is equated to the combined value. In decision box 248, a check is made to determine whether the input signal has been raised to the power of a predetermined exponent. The predetermined exponent is an exponent, g , which is based on an exponent code stored in a control unit of an exponentiation circuit. If the input signal has been raised to the predetermined exponent, the method proceeds to box 250. If not, the method returns to box 242.

In box 250, the exponential signal x^9 is generated by performing an inverse-logarithmic conversion on the combined value.

FIG. 11 illustrates a flow diagram of a method of
5 generating a term which includes a plurality of exponential signals generated from a sequence of input signals. This method may be used in an exponentiation circuit having a sequence of inputs x_i to generate a term having a plurality of exponential signals $x_i^{9^i}$, where i is an integer.

10 In box 260, a first input of the sequence of inputs is received. In turn, in box 262, a binary log value having a plurality of bits is generated by performing a logarithmic conversion on the first input.

In box 264, an exponent code is decoded to produce a
15 first shift signal and a second shift signal. In box 266, the first shift signal is applied to the binary log value to produce a first intermediate value. The first intermediate value may be produced by either shifting the plurality of bits of the binary log value, or by equating
20 the first intermediate value to the binary log value.

In box 268, the second shift signal is applied to the binary log value to produce a second intermediate value. The second intermediate value may be produced by either shifting the plurality of bits of the binary log value; by
25 equating the second intermediate value to the binary log value; or by setting the second intermediate value to zero.

Next, in box 270, the first intermediate value is arithmetically combined with the second intermediate value to produce a combined value. The combined value is
30 produced by either adding the first intermediate value to the second intermediate value, or subtracting the first intermediate value from the second intermediate value. The combined value is then added to a stored value to produce a sum.

35 In decision box 272, a check is made to determine whether all of the sequence of inputs have been received.

If so, the method proceeds to box 276. If not, the method proceeds to box 274.

In box 274, a next input is received from the sequence of input signals. Then, the first input is equated to the
5 next input and the stored value is equated to the sum. The method then returns to box 262.

Finally, in box 276, the term is generated by performing an inverse-logarithm conversion on the sum.

In summary, the present invention as herein-disclosed
10 provides an exponentiation circuit which uses at least one shift register to obtain a power of two multiple of a binary log value. This significantly increases the data throughput rate of the circuit.

Also, the present invention provides an exponentiation
15 circuit which uses an adder circuit, thereby reducing the power and space required by the exponentiation.

A further advantage of the present invention is to provide an exponentiation circuit which uses all digital circuitry, thus making it economical to implement and easy
20 to integrate with a microprocessor core or other digital logic on a single chip.

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
25 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
30 the true spirit and scope of the invention.

What is claimed is:



CLAIMS

1. An exponentiation circuit having an input and an output, the circuit producing an exponential signal on the output in response to receiving an input signal on the input, the circuit comprising:
- 5 a logarithm converter for performing logarithmic conversion on the input signal to produce a binary log value having a plurality of bits;
- 10 shift means for receiving the binary log value and shifting the plurality of bits a distance of n-bits to generate an intermediate value as an output thereof, wherein n is an integer;
- means for arithmetically combining the intermediate value and a stored value to produce a combined value; and
- 15 an inverse-logarithm converter for performing an inverse-logarithmic conversion on the combined value to produce the exponential signal, the inverse-logarithm converter generating the exponential signal on the output of the circuit.
- 20



2. The exponentiation circuit of claim 1, further comprising:

means for storing the binary log value;

a control unit, operatively coupled to the shift

5 means, for applying an exponent code to the shift means;

wherein the shift means generates a sequence of intermediate values in response to a stored binary log value, the shift means shifting the plurality of bits of each of the sequence of intermediate values a bit distance

10 according to the exponent code; and

wherein the combining means includes:

an adder circuit for summing the sequence of intermediate values and the stored value to produce the combined value.

15

3. The exponentiation circuit of claim 2, wherein the control unit converts at least one of the sequence of intermediate values to a signed two's complement form before the at least one of the sequence of intermediate values is received by the adder circuit, the control unit performing the conversion by inverting the at least one of the sequence of intermediate values and providing a corresponding carry bit to the adder circuit.

20

25 4. The exponentiation circuit of claim 1, wherein the shift means includes a shift register.

5. The exponentiation circuit of claim 1, wherein the shift means includes a multiplexer.

30



6. An exponentiation circuit receiving a sequence of input signals and in response producing as output a term which includes a plurality of exponential signals, each of the exponential signals corresponding to one of the input signals, the circuit comprising:

5 a logarithm converter for performing a logarithmic conversion on the sequence of input signals to produce a sequence of binary log values, each of the binary log values having a plurality of bits and corresponding to one of the input signals;

10 a first shift register responsive to the sequence of binary log values and generating a sequence of first intermediate values as an output thereof, the first shift register receiving each of the binary log values and shifting the plurality of bits a distance of n -bits to generate a first intermediate value, wherein n is an integer;

15 a second shift register responsive to the sequence of binary log values and generating a sequence of second intermediate values as an output thereof, the second shift register receiving each of the binary log values and shifting the plurality of bits a distance of n -bits to generate a second intermediate value, wherein n is an integer;

25 a control unit, operatively coupled to the first shift register and the second shift register, the control unit storing a plurality of exponent codes, each of the plurality of exponent codes corresponding to one of the sequence of input signals, the control unit applying the plurality of exponent codes to the first shift register and the second shift register;

30 wherein the first shift register and the second shift register shift the plurality of bits of each of the sequence of binary log values a bit distance according to the corresponding exponent code; and

35



means for arithmetically combining the sequence of first intermediate values and the sequence of second intermediate values to produce a sum; and

- an inverse-logarithm converter for performing an
- 5 inverse-logarithmic conversion on the sum to produce the term, the inverse-logarithm converter providing the term on an output of the exponentiation circuit.



7. The exponentiation circuit of claim 6, wherein the combining means includes:

a first register for providing the sum as an output thereof;

5 a second register for providing a combined value as an output thereof;

a first multiplexer for providing as output either the output of the first register or the output of the first shift register;

10 a second multiplexer for providing as output either the output of the second register or the output of the second shift register;

an adder circuit having a first input responsive to the output of the first multiplexer and a second input responsive to the output of the second multiplexer, the
15 adder circuit generating an output;

de-multiplexing means for providing the output of the adder circuit to either the first register or the second register;

20 wherein the first multiplexer and the second multiplexer are responsive to a mux signal from the control unit and the de-multiplexing means is responsive to a de-mux signal from the control unit;

the control unit asserting the mux signal during a
25 first addition cycle so that the output of the first shift register and the output of the second shift register are provided to the adder circuit;

the control unit asserting the de-mux signal during the first addition cycle so that the output of the adder
30 circuit is provided to the second register;

the control unit asserting the mux signal during a second addition cycle so that the output of the first register and the output of the second register are provided to the adder circuit; and

35 the control unit asserting the de-mux signal during the second addition cycle so that the output of the adder circuit is provided to the first register.



8. The exponentiation circuit of claim 6, wherein the combining means includes:

a first register for providing the sum as an output thereof;

5 a first adder circuit having a first input responsive to the output of the first shift register and a second input responsive to the output of the second shift register, the first adder circuit generating an output;

10 a second adder circuit having a first input responsive to the output of the first adder circuit and a second input responsive to the output of the first register, the second adder circuit generating the sum an output and providing the sum to the first register;

15 wherein the first adder circuit generates a combined value in response to receiving one of the sequence of first intermediate values and one of the sequence of second intermediate values corresponding to the same one of the sequence of input signals; and

20 the second adder circuit generating the sum in response to receiving the combined value from the first adder circuit.



9. A computer system, which comprises:
- a memory for storing a program;
 - an exponentiation circuit having an input and an output, the exponentiation circuit producing an exponential
 - 5 signal on the output in response to receiving an input signal on the input, the exponentiation circuit including:
 - a logarithm converter for performing a logarithmic conversion on the input signal to produce a binary log value having a plurality of bits;
 - 10 a shift register for receiving the binary log value and shifting the plurality of bits a distance of n-bits to generate an intermediate value as an output thereof, wherein n is an integer;
 - means for arithmetically combining the intermediate
 - 15 value and a stored value to produce a combined value; and
 - an inverse-logarithm converter for performing an inverse-logarithmic conversion on the combined value to produce the exponential signal, the inverse-logarithm converter providing the exponential signal on the output of
 - 20 the exponentiation circuit; and
 - a processor, operatively coupled to the memory and to the input and the output of the exponentiation circuit, the processor retrieving the program from the memory and executing the program, and as a result of executing the
 - 25 program, the processor transmitting the input signal to the input of the exponentiation circuit and subsequently retrieving the exponential signal from the output of the exponentiation circuit.



10. In an exponentiation circuit having an input signal x , a method for generating an exponential signal x^g , the method comprising the following steps:

- 5 (a) generating a binary log value having a plurality of bits by performing a logarithmic conversion on the input signal;
- (b) shifting the plurality of bits of the binary log value to generate an intermediate value;
- 10 (c) arithmetically combining the intermediate value and a stored value to produce a combined value, wherein the step of combining is selected from the group consisting of:
- (i) adding the intermediate value to the stored value to produce the combined value; and
- 15 (ii) subtracting the intermediate value from the stored value to produce the combined value; and
- (d) generating the exponential signal by performing an inverse-logarithmic conversion on the combined value.