

AN EFFICIENT BINARY MULTIPLIER WITH REDUCED AREA AND POWER CONSUMPTION**References****US PATENT DOCUMENTS**

US4638449 Multiplier architecture
US5150322 Mixed-radix serial/parallel multipliers
US5262976 Plural-bit recoding multiplier
US4752905 High-speed multiplier having carry-save adder circuit
US5008850 Circuitry for multiplying binary numbers
US5337268 Partial multiplier selector for multiplication circuit
US6446104B1 Double precision floating point multiplier having a 32-bit booth-encoded array multiplier
US5343417 Fast multiplier
US5586071 Enhanced fast multiplier
US5181185 Parallel multiplier using skip array and modified Wallace tree
US5504915 Modified Wallace tree adder for high speed binary multiplier, structure and method
US4901270 Four-to-Two adder cell for parallel multiplication
US5347482 Multiplier tree using nine-to-three adders
US5265043 Wallace tree multiplier array having an improved layout topology
US6029187 Fast regular multiplier architecture

Other publications

A Signed Binary Multiplication Technique, Booth, Andrew Donald, The Quarterly Journal of Mechanics and Applied Mathematics. vol. IV (2): pp 236–240 (1951).
A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations. Kogge, Peter and Stone, Harold. IEEE Transactions on Computers. C-22 (8): 786–793 (1973).

Field of the Invention

The present invention relates to the field of static logic digital circuits. More particularly, the invention relates to an implementation of power and area-efficient binary multipliers by introducing new circuit designs that are used as a drop-in or as a mix-and-match implementations of CMOS binary multipliers.

Background of the Invention

A multiplier is a basic block in many processors. A multiplication circuit or a multiplier consists mainly of three parts: (1) a partial products generator implemented by a plurality of AND or NAND gates, each operating on one bit of a multiplicand and one bit of a multiplier (here, we refer to the multiplying number, as opposed to a circuit), (2) a multiplier array (also called an adder array) made up of columns of adders which reduce the partial products by summation to a “sum” word and a “carry-out” word, and (3) a multi-bit final adder for adding the sum and carry-out words to result in a single output word, i.e. the product. When multiplying two binary numbers, an M-bit multiplicand and an N-bit multiplier, $M \times N$ partial products are usually generated. The resulting product generally has a length of $M+N-1$ binary bits. In most multiplication circuits, both the multiplicand and multiplier are of the same bit length N, and the resultant product is, therefore, $2N-1$ bits long. Any integer number, binary, multiplier comprises multiple summations of multi-bit numbers in a manner similar to decimal integer multiplication.

Multiplication circuits are often constructed with an accumulator to form a multiply-and-accumulate (MAC) processing element or a fused multiply-add (FMA) processing element. These processing elements serve as building blocks in circuits that implement a dot-product of vectors, matrix-vector multiplication, and matrix-matrix multiplication. High-speed processing is required; for example, for artificial intelligence (AI), machine learning (ML), high-performance computing (HPC), and the like that rely on intensive multiplication and addition. Power dissipation is an important figure of merit of a

multiplier. Other important factors are the area and the layout regularity. A regular architecture of a multiplier is easy to route whereas an irregular architecture increases wire congestion and area. The choice of a multiplier's architecture involves trading off between power, area, and delay.

Multipliers are classified by their mode of operation: serial, parallel, or shift-and-add, as well as by their architecture, an array or tree architecture. They are also be classified as unsigned or signed depending on whether they multiply only positive numbers (i.e., unsigned numbers) or signed numbers (i.e., positive and negative numbers). The array architecture of a multiplier is common because of its relatively simple and regular structure. However, array multipliers have higher latency compared to other multiplier architectures. An array architecture has a delay of $O(N)$ (where N is the word length in bits of the operands). It has also, relatively, high switching activity. High switching activity is the primary source of power dissipation of binary multipliers.

Tree architectures are of irregular structure that results in a propagation delay of $O(\log_2 N)$. Tree multipliers are faster than array multipliers but require more complex routing and their area is larger.

Input recoding is one of the most common methods for achieving a compact multiplier. A method of input recoding for reducing the number of partial products was described in "A Signed Binary Multiplication Technique" by Andrew Donald Booth in The Quarterly Journal of Mechanics and Applied Mathematics Vol. IV (2), pp 236–240 (1951). It multiplies two signed binary numbers in two's complement notation as well as a (modified) higher radix to reduce the number of partial products. Booth encoding is commonly used for long operands (of 16 bits or higher) to reduce the number of partial products. However, Booth multipliers trade circuit complexity for speed (i.e., adding an additional circuitry that is used to recode the input and generate the partial products in the Booth encoding scheme).

US patent 4638449 presents an array architecture that relies on a radix-4 Booth encoder for reducing the depth of the multiplier and improving its speed. In patent US5150322, Smith et al. presented a mixed-radix, of 2 and 4, encoders array multiplier for the purpose of saving area and increasing speed. Recoding of two input bits, or more, to reduce the number of partial products was also described in US patent 5262976. A multiplier array structure having a reduced number of summation stages and using a combination of full adders and carry save adders in order to reduce the carry propagation time is described in US4752905. US patent US5008850 described a multiplier that includes a dedicated circuitry for sorting the multiplicand and the multiplying number into groups of contiguous ones and zeros. On top of the common multiplicity of logic AND gates that generate the partial products, the multiplier includes excess circuitry such as a multiplexer as well as a network of multi-input AND gates and OR gates that increases its area as well as its layout complexity. US patent 5,337,268 uses a Booth algorithm multiplier with internal registers that are used to store and sequentially fetch the encoded input to reduce the multiplier latency. US patent 6446104B1 described a double precision floating point multiplier of a 32-bit booth-encoded array multiplier using a partial products accumulator that comprises a carry-save adder, a sum register, a carry-out counter, and an extender.

For tree-architecture, US patents 5,343,417 and 5,586,071 describe a Wallace-tree architecture in which full adders and half adders are used in the multiplier to reduce the partial products by successive addition to sum and carry-out words such that particular inputs to be added at each adder level were designed to increase the multiplier's operating speed. US patent 5,181,185 of Han et al. and US patent 5,504,915 of Rarick disclose modified Wallace-tree adders for summing the partial products aimed at improving speed. Galbi's et al. US patent 4,901,270 presented the use of four-to-two compressor adders in tree multipliers for further improving their speed. In US patent 5,347,482, Williams disclosed the use of nine-to-three adders in a Wallace-tree that

simplifies the layout and routing of the multiplier by using larger basic building blocks in the tree architecture though with no improvement in speed with respect to three-to-two compressor adders. US patent 5,265,043 by Naini et al. discloses a Wallace-tree multiplier architecture that is based on carry-save adders (CSA) arranged in an L-fold floorplan that is aimed to improve the layout's regularity and save area. US patent 6,029,187 describes a multiplier architecture that provides increased operating speed, and yet maintains regularity in its structure to achieve a small floor plan. The multiplier was modified by replacing full adder circuits with compressor circuits in a manner that preserves the balance of the signal propagation delay.

As can be seen, most patents sought to improve the multipliers' power, area, and performance via modifying their architecture (e.g. a tree architecture, high radix, and the like), and rarely via introducing, replacing or modifying the multiplier's standard cells (standard cells are pre-designed and pre-characterized building blocks that are used to implement different functions, i.e. adders, XOR/XNOR gates, etc. in microprocessor design).

It is an object of the present invention to provide an implementation of power and area-efficient binary multipliers by replacing the multiplier's standard cells with more efficient cells.

It is another object of the present invention to provide an implementation of power and area-efficient binary multipliers that apply to any type of multiplier architecture.

It is still another object of the present invention to provide an implementation of power and area-efficient binary multipliers of any bit precision that perform unsigned or signed multiplication with or without input encoding.

Other objects and advantages of the invention will become apparent as the description proceeds.

Summary of the Invention

The present invention relates to static CMOS logic and describes new circuitry for the multiplication of binary numbers.

A signed or unsigned multiplier for the multiplication of binary numbers, comprising a combination of logic cells consisting of standard cells or circuits in any multiplication architecture, which are modified, such that the modified logic cells partially replace CMOS circuits as mix-and-match implementation, or fully replace CMOS circuits as drop-in implementation, in any architecture of the signed or unsigned multiplier.

A final adder of the signed or unsigned multiplier is implemented as mix-and-match or as drop-in of modified standard cells or circuits in any adder architecture.

The significands (in case of a floating-point format) are multiplied using array architecture, a tree architecture, or any other architecture of a multiplier being implemented by the combination of modified logic cells.

The exponents (in case of a floating-point format) are added using a multi-bit adder being implemented by the combination of modified logic cells.

The modified standard cells and circuits may be selected from the group of:

- exclusive-NOR;
- exclusive-OR;
- Selectable logic function gate;
- one-bit half adder;

- one-bit full adder;
- pre-determined bit size adders;
- high fan-in AND gates;
- High fan-in OR gates.

A one-bit half-adder may be dropped-in (i.e., replace a known CMOS standard cell) or mixed-and-matched (i.e., in combination with known CMOS standard cells or circuits) into an array multiplier, the one-bit half-adder is implemented by:

- a) connecting two one-bit inputs to a multiplicity of transistors that compute binary addition;
- b) generating the sum by a first block of the a one-bit half-adder, consisting of a first group of transistors;
- c) generating the Carry out by a second block of the a one-bit half-adder, consisting of a second group of transistors.

A one-bit half-adder may be dropped-in or mixed-and-matched into a tree multiplier; the one-bit half-adder is implemented by:

- a) connecting two one-bit inputs to a multiplicity of transistors that compute binary addition;
- b) generating the sum by a first block of the a one-bit half-adder, consisting of a first group of transistors;
- c) generating the Carry out by a second block of the a one-bit half-adder, consisting of a second group of transistors.

A one-bit half-adder and/or a one-bit full-adder or a one-bit full-adder or their combination are dropped-in or mixed-and-matched into an N-bit adder of any architecture where N is an arbitrary word length, the one-bit half-adder is implemented by:

- a) connecting two one-bit inputs to a multiplicity of transistors that compute binary addition;
- b) generating the sum by a first block of the a one-bit half-adder, consisting of a first group of transistors;
- c) generating the Carry out by a second block of the a one-bit half-adder, consisting of a second group of transistors.

A one-bit half-adder and/or one-bit full-adder or their combination may be dropped-in or mixed-and-matched into a NxM multiplier of any architecture where N and M are arbitrary bit lengths, the one-bit half-adder may be implemented by:

- a) connecting two one-bit inputs to a multiplicity of transistors that compute binary addition;
- b) generating the sum by a first block of the a one-bit half-adder, consisting of a first group of transistors; and
- c) generating the Carry out by a second block of the a one-bit half-adder, consisting of a second group of transistors.

A single-stage high Fan-in logic AND gate may be used for a fast check that the 8-bit exponent of a IEEE-754 floating point 32-bit format or any other integer or floating point binary number format is all-ones for an infinity check as well NaN check, the single-stage high Fan-in logic AND gate is implemented by:

- a) a pull-up stack of n-type transistors that acts as an AND logic gate whose output is fed to a restoration buffer that outputs the result, or,
- b) a CMOS-topology of double-stack of a pull-down n-type transistors network that is connected to a stack of inverters or buffers to compensate for the circuit delay.

A multi-bit adder is implemented using the modified logic cells for adding the exponents.

A single-stage high Fan-in logic NOR gate is implemented by:

- a) a single-stage leakage-tolerant NOR gate which is implemented in tandem with another leakage-tolerant of the NOR gate or a standard CMOS NOR gate, for zero checking, the single-stage leakage-tolerant NOR gate comprises a pull-up stack of p-type transistors, the output of which is fed into a restoration buffer, whose output is the gate's output; and optionally:
- b) an arbitrary number of inputs may be implemented as a pull-up stack of the leakage-tolerant NOR gate where the stack is divided into multiple sub-stacks of a lower number of inputs that are concatenated.

In one aspect, the single-stage high Fan-in logic NOR gate is implemented by a CMOS-topology of double-stack of a pull-up p-type transistors network that is connected to a stack of inverters or buffers to compensate for the circuit delay.

Different bit-lengths of the NOR gates may be implemented for zero checking of an arbitrary bit-length of a floating point or integer binary numbers' format.

An $n+1$ inputs AND gate is used in a carry-skip adder for implementing carry-skip adders wider than 4 bits, where the $n+1$ inputs AND gate may comprise a stack of n-type transistors that acts as an AND logic gate whose output is fed to a restoration buffer that outputs the result. Alternatively, the $n+1$ inputs AND gate may comprise a CMOS-topology double-stack of a pull-down n-type transistors network that is connected to a stack of inverters or buffers to compensate for the circuit delay.

An arbitrary combination of the modified half-adder and full adders may be dropped-in or mixed-and-matched with known-CMOS full and half adders.

A selectable logic gate, executing a pre-selected logic function, comprising:

- a) a first inverter, the output of which is being connected to the output of the selectable logic gate;
- b) a pair of parallel-connected complementary transistors, having common source and drain ports and separate gating inputs, the common drain and source ports being the output of the selectable logic gate; and
- c) a plurality of inputs that configure the selectable logic gate, to execute a specific logic function.

The selectable logic gate may be configured to a pre-selected logic function.

Unless otherwise defined, all technical and/or scientific terms used herein have the same meaning as commonly understood by one of ordinary skill in the art to which the invention pertains. Although methods and materials similar or equivalent to those described herein can be used in the practice or testing of embodiments of the invention, exemplary methods and/or materials are described below. In case of conflict, the patent specification, including definitions will control. In addition, the materials, methods, and examples are illustrative only and are not intended to be necessarily limiting.

Brief Description of the Drawings

The above and other characteristics and advantages of the invention will be better understood through the following illustrative and non-limitative detailed description of preferred embodiments thereof, with reference to the appended drawings, wherein:

Fig. 1a depicts a full voltage swing two-input XOR gate implementation of the present invention comprising a pair of parallel-connected complementary transistors, having common source and drain ports and separate gating inputs, the common drain and source ports connected the output inverter and a two-transistors pull-up (p-type) and pull-down (n-type) transistors;

Fig. 1b depicts another full voltage swing, two-input XOR gate implementation of the present invention;

Fig. 2 depicts a full-voltage swing, two-input selectable logic gate of the present invention, executing a pre-selected logic function;

Fig. 3 is a circuit diagram of a one-bit half adder of the present invention;

Fig. 4a (prior art) is a circuit diagram of a one-bit full adder;

Fig. 4b (prior art) is a circuit diagram of a one-bit full adder with inverted outputs;

Fig. 5a is a leakage-tolerant single-stage high Fan-in logic (10 inputs AND) gate of the present invention comprising a pull-up stack of n-type transistors that acts as an AND logic gate whose output is fed to a restoration buffer that outputs the result;

Fig. 5b is a single-stage implementation of a high Fan-in logic (10 inputs AND) gate of the present invention comprising a CMOS-topology, double-stack, of a pull-down n-type transistors network that is connected to a stack of inverters or buffers to compensate for the circuit delay;

Fig. 6a is a leakage-tolerant single-stage 8-input NOR gate of the present invention comprising a plurality of p-type transistors pull-up stack, the output of which is fed into a restoration buffer, whose output is the gate's output;

Fig. 6b is a single-stage implementation of a high Fan-in logic (8 inputs NOR) gate of the present invention comprising a CMOS-topology double-stack of a pull-up p-type transistors network that is connected to a stack of inverters or buffers to output the result;

Fig. 7a is a prior art of circuit diagrams of a radix-4 (also termed as modified Booth decoder). Booth's decoder circuit (left) and Booth's encoder circuit (right);

Fig. 7b is a circuit diagram of a radix-4 Booth decoder of the present invention;

Fig. 8 is a block diagram of a public domain n-bit carry-skip adder;

Fig. 9 is a block diagram of a public domain 4 bits carry-ripple (also termed ripple-carry) adder;

Fig. 10 is a block diagram of a public domain 4-bit carry-save adder;

Fig. 11 is a block diagram of a public domain implementation of a 4x4 unsigned A and B binary numbers array multiplier where the partial products are $p_{ij} = A_i B_j$.

Fig. 12 is a block diagram of a public domain implementation of a 4x4 unsigned A and B binary numbers (Wallace) tree multiplier where the partial products are $P_{ij} = A_i B_j$.

Fig. 13 is a block diagram of a prior art implementation of an 8-bit Kogge-Stone parallel prefix adder of A and B binary numbers where $p_i = A_i \oplus B_i$ and $g_i = A_i B_i$ are the propagate (p_i) and generate (g_i) signals and the black square represent Kogge-Stone's black cells that generate the groups' propagate and generate signals P_{ij} and G_{ij} ; and

Fig. 14 is a simplified block diagram of a public domain (IEEE-754) floating point binary numbers multiplier architecture where zero check, infinity check and NaN check circuits have been omitted for clarity.

Detailed Description of the Present Invention

The present invention relates to an implementation of power and area-efficient binary multipliers by introducing new circuit designs. The disclosed embodiments present new static-logic circuits that are used as a drop-in implementation in CMOS binary multipliers (where CMOS circuits or standard cells are fully replaced by modified logic cells which are power and area-efficient) or as a mix-and-match implementation in CMOS binary multipliers (where CMOS circuits or standard cells are partially replaced by modified logic cells which are power and area-efficient).

The methods and circuits that are described in the present invention replace (i.e. as a drop-in or mix-and-match) a multiplier's standard cells with more efficient cells. These methods apply to any type of multiplier architecture including but not limited to parallel and sequential multipliers, pipelined and systolic arrays of processing elements that include multipliers, circuits for computing elementary functions, etc. The above-referenced multiplier methods apply for any bit precision, unsigned, or signed multiplication with or without input encoding.

Figs. 1a presents a full-voltage swing two-input XOR gate circuit, according to an embodiment of the present invention. The two-input XOR gate circuit comprises a transmission gate comprising a pair of parallel-connected complementary transistors, having common source and drain ports and separate gating inputs and a pMOS-nMOS transistors stack to save power, and area and improve performance in comparison with known CMOS XOR gate circuits.

Input A is fed to an inverter and input B is fed to another inverter. The inverted B input is fed to the transmission gate at 1 and the pMOS-nMOS stack pin 5 and 6. The inverted input A and input A are fed to the control pins 3, 2 of the transmission gate, respectively. The output pin 8 generates the exclusive-or operation $A \oplus B = \bar{A}B + A\bar{B}$. Alternatively, the same circuit, with one of the inputs inverted (i.e. the inputs are $A, \bar{B}$ or $\bar{A}, B$ by dropping one of the input inverters) implements a two-input XNOR gate. For example, inverting the input B of Fig. 1a (to $\bar{B}$) results in an XNOR gate that is more power efficient than known CMOS. In an embodiment of the present invention different combinations of $A, B, \bar{A}, \bar{B}$ implement XOR or XNOR circuits that are more power efficient than known CMOS. Yet in another embodiment of the present invention, the output of the circuit is fed to an inverter, to a buffer or a chain of inverters, or to a stack of inverters or their combinations to implement XOR or XNOR circuits of different drive currents.

Fig. 1b presents another full-voltage swing two-input XOR gate embodiment of the present invention to save power. The output pin 9 generates the exclusive-or operation $A \oplus B = \bar{A}B + A\bar{B}$.

Alternatively, the same circuit, with one of the inputs inverted (i.e. $A, \bar{B}$ or $\bar{A}, B$ by dropping one of the input inverters) implements a two-input XNOR gate. For example, inverting the input B (to $\bar{B}$) results in an XNOR gate that is more power efficient than

known CMOS. In an embodiment of the present invention different combinations of $A, B, \bar{A}, \bar{B}$ implement XOR or XNOR gate circuits that are more power efficient than known CMOS. Yet in another embodiment of the present invention the output of the circuit is fed to an inverter, to a buffer, to a chain of inverters or a stack of inverters or buffers or their combinations to implement XOR or XNOR circuits of different drive currents.

Fig. 2 presents a full-voltage swing, two-input, selectable logic gate standard cell embodiment of the present invention. The circuit generates different logic functions at the output Z depending on the inputs', X1...X6, configuration. Inputs A and B or their complementary signals are fed into different X_i pins to implement a pre-selected logic function.

Table I depicts several two-input configurations that implement different logic functions (e.g., XOR, AND, NOR, OR, NAND).

Table I

X1	X2	X3	X4	X5	X6	Z
B	A	$\bar{A}$	B	A	$\bar{A}$	$A \oplus B$
$\bar{B}$	A	0	A	1	B	AB
B	$\bar{A}$	0	0	1	A	$\overline{A + B}$
$\bar{A}$	A	B	B	A	0	$A + B$
B	1	$\bar{A}$	$\bar{A}$	$\bar{B}$	B or 0	$\overline{AB}$

Fig. 3 presents a circuit diagram of a one-bit half-adder embodiment of the present invention. Two one-bit numbers A and B are fed into terminals 10 and 11, respectively. The sum $A \oplus B$ is generated at terminal 12 and the carry-out, $A * B$, at terminal 13. In an embodiment of the present invention, this one-bit half-adder is used for column reduction in a tree-multiplier via the known Wallace or Dadda reduction schemes or any other reduction scheme.

In another embodiment, the half adder is dropped into a multi-bit carry-skip adder, carry-ripple adder or carry-save adder or into any bit-length multi-bit adder of any architecture. Yet in another embodiment of the present invention, the half-adder is used to sum partial products or elements in any multiplier of any bit-length and of any architecture.

In another embodiment, a modified one-bit half-adder is implemented by connecting two one-bit inputs to a multiplicity of gates of single-type transistors that compute binary addition and generating the sum by a first block of the a one-bit half-adder, which consists of a first group of transistors. Then, the Carry out is generated by a second block of the one-bit half-adder, consisting of a second group of transistors. Finally, the modified one-bit half-adder is dropped-in or mixed and matched into an array multiplier or into a tree multiplier.

In another embodiment, modified one-bit half-adder or a modified one-bit full-adder are implemented by connecting two one-bit inputs to a multiplicity of gates of single-type transistors that compute binary addition and generating the sum by a first block of the a one-bit half-adder, which consists of a first group of transistors. Then, the Carry out is generated by a second block of the one-bit half-adder, consisting of a second group of transistors. Finally, the modified one-bit half-adder and/or a modified one-bit full-adder or their combination are dropped-in or mixed and matched into an N-bit adder of any architecture, or into a NxM multiplier of any architecture where N and M are arbitrary.

Fig. 4a (prior art) is a circuit diagram of a one-bit full adder of known CMOS comprising 28 transistors. Two one-bit numbers A (14) and B (15) are fed respectively with a Carry-in (16). The sum, $A \oplus B \oplus C_{in}$ is generated at 17 and the Carry-out, $A * B + C_{in}(A \oplus B)$, is generated at 18.

Fig. 4b (prior art) is a circuit diagram of a one-bit full adder that is presented in Fig. 4a with inverted outputs. Namely, this adder generates an inverted SUM ($\overline{\text{SUM}}$, 19) and inverted Carry-out ($\overline{\text{Cout}}$, 20). In an implementation of the present invention, the inversion property of a one-bit full adder is implemented in a multiplier via the inverted output one-bit full adder with any of the two-input XOR gates of the present invention, to realize a selectable logic function gate. The selectable logic function gate is configured as a two-input XOR, or the half adder of the present invention or their combination to save power and area.

In one embodiment of the present invention, any XOR/XNOR gate of the present invention is used to replace the known CMOS XOR/XNOR gates in the known CMOS full adders to save the power and area of a multiplier. Yet in another embodiment of the present invention, two XOR or XNOR gates of the present invention are used as replacements for the XOR gates of the known CMOS full-adders to save power and area.

Yet in another embodiment of the present invention, a modified known CMOS one-bit full adder according to the present invention is dropped-in or mixed and matched into a multi-bit adder to save power and area. In another embodiment of the present invention, a modified know CMOS one-bit full-adder (that is implemented with any of the XOR or XNOR gates of the present invention) is dropped-in or mixed and matched into a multiplier to save power and area.

Fig. 5a is a single-stage leakage tolerant high Fan-in logic AND gate of the present invention comprising a pull-up stack of n-type transistors that acts as an AND logic gate whose output is fed to a restoration buffer that outputs the result. In an embodiment of the present invention, ten data inputs A1 to A10 are fed to a stack of n-type transistors, 21, that acts as an AND logic gate whose output is fed to a restoration buffer 22 that outputs the result to 23. Inverting the output of the gate results in a single-stage high Fan-in logic NAND gate. In an embodiment of the present invention such high-fan in

gate is implemented as a replacement for a tree-of-gates of known CMOS to improve power consumption and save power.

IEEE-754 32bit floating point format conditions of Zero, Infinity, and NaN numbers are given in Table II below.

Table II

Exponent (8 bit)	Mantissa (23 bit)	Value
0	0	Exact Zero
255	0	Infinity
255	Not 0	Not a Number (NaN)

Zero, Infinity, and Not a Number (NaN) checking are carried out prior to any multiplication operation. In an embodiment of the present invention an 8-input single stage leakage-tolerant AND gate is used for a fast check that the 8-bit exponent of a floating point 32-bit format (FP32) is all-ones for an infinity check as well as NaN check. In another embodiment of the present invention’s leakage-tolerant AND gate; an optimal leakage-tolerant operation is achieved by placing the gate in circuits in which the output of logic “1” is, relatively, rare (e.g., Infinity or NaN values). Therefore, a time average (of the inputs fed into the gate) or a statistically-averaged power consumption that scales down as $\frac{1}{2^N}$ (in the case of uniformly distributed inputs, where N is the number of inputs, e.g. N = 8, 10 or higher, results in significant power saving in comparison with known CMOS equivalent implementation of the gate as well as (up to 200x faster) faster operation than checking by software. In another embodiment of the gate, an 11-inputs single-stage leakage-tolerant AND gate of the present invention is implemented for Infinity and NaN check of IEEE-754 64-bit format (double precision). Yet, in other embodiment, the high fan-in single-stage leakage-tolerant AND gate implements Infinity and NaN check in multipliers of other formats such as BFLOAT-16

format (8-bit exponent), NVidia's 16-bit TensorFloat format (8-bit exponent) or 32-bit TensorFloat format (8-bit exponent), AMD's fp24 format (7-bit exponent) and the like.

With reference to Fig. 5b, a static CMOS high Fan-in logic AND gate (i.e. 10 inputs) of the present invention is depicted comprising A double stack of a pull-down network (PDN, 24) that is used to overcome parasitic resistance and capacitance as well as an inverters-stack, 25, at the output to compensate for the RC-delay of the PDN stack. In an embodiment of the present invention such high-fan in gate is implemented as a replacement for a tree-of-gates of known CMOS to improve power consumption and save power. In another embodiment of the present invention the high Fan-in gate is used to implement an Infinity and NaN checks as was described earlier. In another embodiment of the present invention multiple stacks of PDNs of different combinations (series, parallel and series-parallel) are implemented to compensate for RC-delay. Yet in another embodiment of the present invention, other stack combinations of inverters or buffers (series, parallel and series-parallel) are used at the output 25 to compensate for RC-delay or control the drive current of the gate.

With reference to Fig. 6a, an implementation of a single-stage leakage-tolerant 8-input NOR gate comprises a pull-up stack of p-type transistors, the output of which is fed into a restoration buffer, whose output is the gate's output that overcomes the 4-input limitation or tree-of-gates implementation of known CMOS standard cells to save power and area.

Data inputs A1 to A8 are fed to a stack of p-type transistors, 26, that acts as NOR logic gate. The output of the stack is fed into a restoration buffer, 27, whose output, 28, is the gate output. Inverting the output, 28, results in an OR logic gate.

In one embodiment of the leakage-tolerant NOR gate; a 23-input NOR gate comprising twenty-three p-type transistors-stack is implemented in tandem with an 8-bit NOR gate

implementation the gate for zero checking according to IEEE-754 32-bit floating point format to save power and area. Inverting the output of the gate results in an OR gate. In another embodiment of the present invention, an arbitrary number of inputs is implemented as a p-type transistors-stack leakage-tolerant OR gate or NOR gate where the stack is divided into multiple sub-stacks of a lower number of inputs that are concatenated. In another embodiment of the gate, different bit-length OR gates or NOR gates are implemented for zero checking of an arbitrary bit-length floating point or integer binary number format. The use of the high Fan-in leakage-tolerant NOR gate of is limited in some applications (e.g. sparse matrix multiplication).

Fig. 6b presents a static CMOS circuit of a high Fan-in NOR gate of the present invention that overcomes the 4-input limitation of known CMOS to save power and area. A double stack pull-up network (PUN, 29) is implemented to reduce the RC-delay as well as an output stack of inverters, 30, to improve speed and control the driving current of the gate. Inverting the output results in a high Fan-in OR gate. In an embodiment of the present invention, the gate is used in the same way as was described for the high Fan-in leakage-tolerant NOR gate of the present invention.

A double stack intertwined-topology, 29-30, is used to reduce the RC delay of the stack and facilitate a high number of inputs (i.e. more than 4 inputs of known CMOS) to provide a single stage 8-input NOR gate instead of the tree-topology that is used to implement such high fan-in gate in known CMOS.

Fig. 7a is a prior art of a modified (Radix-4) Booth encoder and decoder circuits. A Radix-4 encoder circuit, 31, is depicted that outputs signals Neg, X1, Z, and X2 to encode the multiplier (to a Radix-4 representation). The encoded multiplier and the multiplicand are fed into the decoder, 32, to generate a partial product (p_{ij}). In an embodiment of the present invention the XOR and XNOR gates of the encoder are replaced by any of the XOR and XNOR gates of the present invention. In another embodiment of the present

invention, any of the XNOR gates of the modified Booth decoder is replaced with an XNOR gate of the present invention.

Fig. 7b is a circuit diagram of a modified Booth decoder implementation, 33, of the present invention to save power and area. The encoded multiplier and the multiplicand are fed into the decoder to generate the partial product (p_{ij}). In another embodiment of the present invention the XOR gate of the decoder is replaced by any of the XOR gates of the present invention.

Fig. 8 (prior art) depicts a block diagram of n-bit carry-skip (CSkA) adder. Two n-bit numbers A_n and B_n are added as well as Carry-in 34. The result is the list of numbers S_0 (LSB, 35) to S_n (MSB, 36) and carry-out at the output of the OR gate, 38. a square designated as FA is a one-bit full adder cell. A carry skip sub-circuit comprises an n+1 inputs AND gate, 39, that is fed by n+1 partial products $p_i = A_i B_i$.

In one embodiment of the present invention, the one-bit full adders of known CMOS in a CSkA are replaced by a modified single or multiple modified one-bit full adders according to the present invention thereby saving power and area as described earlier. In another embodiment of the present invention, an n+1 input high Fan-in AND gate of the present invention is used in the carry-skip adder, 39, instead of the known CMOS AND gate. Yet, in another embodiment of the present invention, a single-stage high fan-in AND gate of nine inputs exceeding that of known CMOS (any single-stage CMOS standard cell is limited to four inputs) is dropped-in to implement a single-block carry-skip adder of 8-bits (beyond known CMOS's 4-bit wide carry skip adder), thereby improving performance, power, and area. Yet in another embodiment, any of the high Fan-in AND gates the present invention is used to implement an arbitrary bit long single-block carry-skip adder. In another embodiment, any mix-and-match combination of known CMOS full adders of that are implemented with the present invention's XOR or

XNOR gates is implemented to improve power consumption, save area and increase the speed of any multi-bit adder.

Fig. 9 presents a prior art 4-bit Carry-Ripple Adder (CRA). Two 4-bit numbers, A and B are added as well as carry-in 40. FA cell designates a one-bit full adder cell. Output 41 is the carry-out of the summation. The list of the sums S0 (LSB) to S3 (MSB) is the addition's result.

In an embodiment of the present invention, a modified one-bit full-adder is implemented with an XOR or XNOR gates of the present invention or their combination and is dropped into a CRA instead of a single or plurality of known CMOS one-bit full adders or mixed-and-matched with known CMOS one-bit full adders for saving power and area. In an embodiment where the least-significant bit is implemented by a half-adder (i.e., no carry-in), the half-adder of the present invention is dropped in as a substitute for known CMOS one-bit half-adder to save area and power. In another embodiment, any mix-and-match combination of the prior art full adders is implemented with XOR or XNOR gates and/or the half-adder of the present invention to improve power consumption, save area and increase the speed of a CRA of any bit length.

With reference to Fig. 10, the prior art of a multi-bit adder carry save adder (CSA) is presented for the addition of two 4-bit binary numbers A and B, and carry-in (42). The list of numbers S0 (LSB) to S3 (MSB) and a carry-out, 43, provide the result.

In an embodiment of the present invention, an arbitrary combination of the half-adder of the present invention and modified full adders with XOR or XNOR gates of the present invention are dropped in or mixed-and-matched with known CMOS full and half adders in any bit-length carry-save adder to save power and area. Yet in another

embodiment, the XOR implementation of the present invention is dropped in instead of the known CMOS XOR gate, 44, to save power and area.

Fig. 11 presents a prior art architecture of an unsigned array multiplier to illustrate the multiplication of two 4-bit numbers $A \times B$. The partial products $A_i B_j \stackrel{\text{def}}{=} p_{ij}$ are generated by either two-input AND or two-input NAND gates and are fed into a rectangular array of addition elements (full adders and half adders) with the output product comprising $2N - 1$ (i.e. bits F0 to F6 in Fig. 11) bits and a Carry out, F7. The partial product $p_{00} = A_0 B_0 \equiv F_0$, 45, and so for the rest. HA, 46, stands for half adder and the FA, 47, for a full adder. The multiplier results in a list of numbers F0 (LSB) to F6 (MSB).

In an embodiment of the present invention, an arbitrary combination of the present invention's half-adder and modified full adders with XOR or XNOR gates of the present invention; are dropped in or mixed-and-matched in any known CMOS adders to implement any array multiplier to save power and area.

With reference to Fig. 12, a prior art implementation of a (Wallace) tree multiplier is presented for the multiplication of two 4-bit numbers, $A \times B$. The partial products $A_i B_j \stackrel{\text{def}}{=} p_{ij}$ are fed to a tree array of addition elements (full adders and half adders) with the output product comprising $2N - 1$ bits and a Carry out bit. The partial product $p_{00} = A_0 B_0 \equiv F_0$, 48, and so for the rest. HA, 49, is a half adder, and FA, 50, is a full adder. The multiplier results in a list of numbers F0 (LSB) to F6 (MSB). The output F7 is carry-out.

A tree multiplier is faster than an array multiplier and is commonly used; along with more sophisticated partial product generation schemes (i.e. Booth encoding) that reduce the number of partial products to less than N^2 as described in US patents 4638449, 5150322, 5262976 and others. According to the present invention, a

combination of the present invention's half-adder and modified full adders implemented with XOR or XNOR gates of the present invention are dropped in or mixed and matched with known CMOS adders to implement any tree multiplier to save power and area.

Fig. 13 is a prior art example of a parallel prefix adder based on the Kogge-Stone algorithm to add two n-digit numbers A and B (8-bit numbers in Fig. 13). The terms $g_i \equiv A_i B_i$ and $p_i \equiv A_i \oplus B_i$ are carry-generate and carry-propagate respectively. The black cells are used to output the generalized-generate and generalized-propagate terms $G_i = g_i + p_i g_{i-1}$ and $P_i = p_i p_{i-1}$. In an embodiment of the present invention, a XOR gate of the present invention is used to implement P_i and as a drop in for p_i in G_i .

Fig. 14 presents a simplified public domain block diagram of one implementation of a multiplier of two 32-bit floating point numbers. Each of the input numbers A and B are separated into a sign, 51, exponent, 52, and significand, 53. Multiplying two numbers results in a negative sign number if one of the numbers is of a negative value. An XOR gate (51) is used for the determination of the sign bit of the product. For the addition of exponents; an unsigned adder is used. For significands multiplication; an unsigned multiplier is used with or without input encoding.

In one embodiment of the present invention, an XOR circuit of the present invention is used for a sign calculation for saving power and area. Yet in another embodiment of the present invention, 8-bit long or arbitrary concatenation of a single type of adder (e.g. a 3-bit adder concatenated with a 5-bit adder) or any combination of different bit lengths and different types of adders according to the present invention is implemented for exponents addition using the methods that were described before. In another embodiment, any bit long adder is implemented with the drop-in or mix-and-match methods, that have been described, for adding the exponents. Without limiting the scope of the present invention, other prior art architectures of multi-bit adders such as

Carry look-ahead, Kogge-Stone, Brent-Kung, Ladner-Fischer, Han-Carlson, Beaumont-Smith, Jackson, Ling, Krapchenko, Sklansky, Knowles and others can be implemented with the drop-in and mix-and-match methods of the present invention in any multiplier architecture.

In another embodiment of the present invention, the significands (the mantissa of a floating-point number) of a floating point number are multiplied, 53, using an array or tree or any other architecture of a multiplier according to the drop-in and mix-and-match methods of the present invention as was described for multipliers.

Yet in another embodiment of the present invention, whenever the last part of a signed or unsigned multiplier comprises a final adder to generate in the final product; the drop-in and mix-and-match methods of the present invention that were described for multi-bit adders can be implemented for any multi-bit adder as well as for an arbitrary concatenation of multiple single-type or different types of adders (e.g. k-bit long Carry-skip adder concatenated with l-bit of Carry-ripple adder concatenated with m-bit Carry-save adder and so on and forth) to create an adder of any bit-length and placement-order as well as any arbitrary combination of sub-adders.

The methods and circuits described in the present invention apply to any bit width and any type of multiplier architecture including but not limited to parallel and sequential multipliers, pipelined and systolic arrays of processing elements that include multipliers, circuits for computing elementary functions, etc.

It is appreciated that certain features of the invention, which are, for clarity, described in the context of separate embodiments, may also be provided in combination in a single embodiment. Conversely, various features of the invention, which are, for brevity, described in the context of a single embodiment, may also be provided separately or in any suitable sub-combination or as suitable in any other described embodiment of the invention. Certain features described in the context of various embodiments are not to

be considered essential features of those embodiments, unless the embodiment is inoperative without those elements.

Although the invention has been described in conjunction with specific embodiments thereof, it is evident that many alternatives, modifications and variations will be apparent to those skilled in the art. Accordingly, it is intended to embrace all such alternatives, modifications, and variations that fall within the spirit and broad scope of the appended claims.

Claims

1. A signed or unsigned multiplier for the multiplication of binary numbers, comprising a combination of logic cells consisting of standard cells or circuits in any multiplication architecture, which are modified, such that said modified logic cells partially replace CMOS circuits as mix-and-match implementation, or fully replace CMOS circuits as drop-in implementation, in any architecture of said signed or unsigned multiplier.
2. A multiplier according to claim 1, in which a final adder of the signed or unsigned multiplier is implemented as mix-and-match or as drop-in of modified standard cells or circuits in any adder architecture.
3. A multiplier for floating point numbers according to claim 1, in which the significands are multiplied using array architecture, a tree architecture, or any other architecture of a multiplier being implemented by the combination of modified logic cells.
4. A multiplier for floating point numbers in which the exponents are added using a multi-bit adder being implemented by the combination of modified logic cells.
5. A multiplier according to claim 1-4, in which the standard cells and circuits are selected from the group of:
 - exclusive-NOR;
 - exclusive-OR;
 - Selectable logic function gate;
 - one-bit half adder;
 - one-bit full adder;
 - pre-determined bit size adders;
 - high fan-in AND gates;

- High fan-in OR gates.
6. A multiplier according to claim 2, in which a one-bit half-adder is dropped-in or mixed and matched into an array multiplier, said one-bit half-adder is implemented by:
- a) connecting two one-bit inputs to a multiplicity of gates of single-type transistors that compute binary addition;
 - b) generating the sum by a first block of said one-bit half-adder, consisting of a first group of transistors;
 - c) generating the Carry out by a second block of said one-bit half-adder, consisting of a second group of transistors.
7. A multiplier according to claim 2, in which a one-bit half-adder is dropped-in or mixed-and-matched into a tree multiplier, said one-bit half-adder is implemented by:
- d) connecting two one-bit inputs to a multiplicity of gates of single-type transistors that compute binary addition;
 - e) generating the sum by a first block of said one-bit half-adder, consisting of a first group of transistors;
 - f) generating the Carry out by a second block of said one-bit half-adder, consisting of a second group of transistors.
8. A multiplier according to claim 2, in which a modified one-bit half-adder and/or a modified one-bit full-adder or their combination are dropped-in or mixed-and-matched into an N-bit adder of any architecture where N is arbitrary, said modified one-bit half-adder is implemented by:
- a) connecting two one-bit inputs to a multiplicity of gates of single-type transistors that compute binary addition;

- b) generating the sum by a first block of said one-bit half-adder, consisting of a first group of transistors;
 - c) generating the Carry out by a second block of said one-bit half-adder, consisting of a second group of transistors.
9. A signed or unsigned multiplier according to claim 2, in which a modified one-bit half-adder and/or a modified one-bit full-adder or their combination are dropped-in or mixed-and-matched into a NxM multiplier of any architecture where N and M are arbitrary, said modified one-bit half-adder is implemented by:
- a) connecting two one-bit inputs to a multiplicity of gates of single-type transistors that compute binary addition;
 - b) generating the sum by a first block of said one-bit half-adder, consisting of a first group of transistors;
 - c) generating the Carry out by a second block of said one-bit half-adder, consisting of a second group of transistors.
10. A multiplier according to claim 3, in which a single-stage high Fan-in logic AND gate is used for a fast check that the 8-bit exponent of a floating point 32-bit format or any other integer or floating-point binary number format is all-ones for an infinity check as well NaN check, said single-stage high Fan-in logic AND gate is implemented by:
- a) a stack of n-type transistors that acts as an AND logic gate whose output is fed to a restoration buffer that outputs the result, or,
 - b) a CMOS-topology double stack of a pull-down n-type transistors network, to compensate for output delay.

11. A multiplier according to claim 3, in which a multi-bit adder is implemented using the modified logic cells for adding the exponents.
12. A multiplier according to claim 3, in which a single-stage leakage-tolerant NOR gate is implemented in tandem with another leakage-tolerant of said NOR gate or a CMOS-topology of double-stack of a pull-up p-type transistors network that is connected to a stack of inverters or buffers, or standard CMOS NOR gate, for zero checking, said single-stage leakage-tolerant NOR gate comprises a stack of p-type transistors, the output of which is fed into a restoration buffer, whose output is the gate output.
13. A multiplier according to claim 5, in which an arbitrary number of inputs is implemented as a p-type transistors-stack of the leakage-tolerant NOR gate where the stack is divided into multiple sub-stacks of a lower number of inputs that are concatenated, or,
a CMOS-topology NOR gate comprising a double-stack of a pull-up p-type transistors network that is connected to a stack of inverters or buffers to compensate for the circuit delay.
14. A multiplier according to claim 12, in which different bit-length of said NOR gates are implemented for zero checking of an arbitrary bit-length floating point or integer binary number format.
15. A multiplier according to claim 2, in which an $n+1$ inputs high Fan-in AND gate is used in a carry-skip adder, implementing carry-skip adders wider than 4 bits, where said $n+1$ inputs high Fan-in AND gate comprises a stack of n-type transistors that acts as an AND logic gate whose output is fed to a restoration buffer that outputs the result or, said $n+1$ inputs high Fan-in AND gate comprises a double stack of a

pull-down network that is used to overcome parasitics to compensate for output delay.

16. A multiplier according to claims 1-4, in which an arbitrary combination of the modified half-adder and full adders are dropped-in or mixed-and-matched with known-CMOS full and half adders.
17. A selectable logic gate, executing a logic function, comprising:
 - a) a first inverter, the output of which is being connected to the output of said selectable logic gate;
 - b) a pair of parallel connected complementary transistors, having common source and drain ports and separate gating inputs, said common drain and source ports being the output of said selectable logic gate; and
 - c) a plurality of inputs that configure said selectable logic gate, to execute a specific logic function.
18. A selectable logic gate according to claim 16, configured as a XOR gate.
19. A selectable logic gate according to claim 16, used for sign checking.