COMPLEX NOISE-BITS AND LARGE-SCALE INSTANTANEOUS PARALLEL OPERATIONS WITH LOW COMPLEXITY

HE WEN[a, b], LASZLO B. KISH[a], AND ANDREAS KLAPPENECKER[c]
[a]*Texas A&M University, Department of Electrical and Computer Engineering, College Station, TX 77843-3128, USA*
[b]*Hunan University, College of Electrical and Information Engineering, Changsha, 410082, China*
[c]*Texas A&M University, Department of Computer Science, College Station, TX 77843-33112, USA*

Received ()
Revised ()
Accepted ()
Communicated by

We introduce the complex noise-bit as information carrier, which requires noise signals in two parallel wires instead of the single-wire representations of noise-based logic discussed so far. The immediate advantage of this new scheme is that, when we use random telegraph waves as noise carrier, the superposition of the first 2^N integer numbers (obtained by the Achilles heel operation) yields non-zero values. We introduce basic instantaneous operations, with $O(2^0)$ time and hardware complexity, including bit-value measurements in product states, single-bit and two-bit noise gates (universality exists) that can instantaneously operate over large superpositions with full parallelism. We envision the possibility of implementing instantaneously running quantum algorithms on classical computers while using similar number of classical bits as the number of quantum bits emulated without the necessity of error corrections. Mathematical analysis and proofs are given.

Keywords: Noise-based logic; noise-based computations; deterministic logic.

1. Introduction

Very recently, new types of deterministic logic systems and computing have been introduced, noise-based logic, where the information carrier is a random noise [1-9]. The bit values can be represented by various types of independent (orthogonal) random noises, their products (including their set-theoretical products for brain logic), and their superposition [1]. The independent base noises and their superpositions form the logic base and its space and the products of independent noise form the logic hyperspace [1]. Generally, from N independent base noises, which are called noise-bits, a hyperspace of 2^N dimensions can be formed, which means 2^{2^N} logic values when superpositions are utilized [2,3].

Noise-based logic versions have been classified according to their signals (*continuum* [1,2], *spikes* [3-5], *random-telegraph waves* [4-7], etc.), or their way of evaluating logic values (*correlator-based* [1,2] or *instantaneous* [4,5]). All these logic schemes have been proven universal [1,4,5] when they represent Boolean logic variables.

Also an efficient string verification algorithm [7] has been introduced that is utilizing product vectors in the hyperspace while computing the hash function during its operation.

In the present paper, we shall introduce a new instantaneous logic scheme, which corresponds to the universal logic system of quantum computers while using less hardware and time complexity for operations. Some of the advantages of the scheme that it is free from decoherence-related errors and it can be realized by today's technology. In Section 2, we introduce the concepts and illustrate them

2. The earlier, real number based RTW representation

In the present paper, we will utilize discrete-time, two-state random telegraph waves (RTW) [4,5,7] , which are two-state random noises to identify the noise-bit and its value. This is not strictly necessary however it has practical advantages during practical realization. Our RTWs have either +1 or -1 value, which is randomly selected with 50% probability at the beginning of each clock period, and this value does not change during the rest of the clock interval. Two product of independent RTWs will be a new RTW that is statistically independent from the original ones thus the product does not require higher bandwidth than the original RTWs do [7].

Let the stochastic time functions $A_k \equiv A_k(t)$ and $B_k \equiv B_k(t)$ $(k = 1,...n)$ be independent, discrete-time RTW signals with ±1 random values, each value with 50% probability. In the earlier, real number based RTW representation, the *k*-th noise bit and its values are defined as:

$$\left|0_k\right\rangle = A_k \quad \text{and} \quad \left|1_k\right\rangle = B_k \tag{1}$$

The noise representing the product state $\left|x_1 x_2 ... x_n\right\rangle = X_1 X_2 ... X_n$ (bit string) of *n* noise-bits (where x_m is the actual noise-bit value, 0 or 1, and X_m is its RTW) will also be an RTW, which is statistically independent from any of the RTWs in the product [7]. Such a representation can be utilized for hashing and efficient string verification [7]. It is a great advantage of this system that the *k*-th noise-bit's logic value can instantaneously be inverted (NOT operation) by multiplying the product by $A_k B_k$ and this holds even for the superposition of such product strings, indicating a large parallelism. Similar other operations are also possible by multiplying the system state with a polynomial expression of the RTWs representing noise bits, see Section 3. However, measurements of bit values in large products and interacting bit operations need the use of "ghost" states which are the most powerful tools for large parallelism, see Section 2.3.

However, the computing potential of this scheme is greatly reduced when we generate the superposition $Y(2^n)$ of the first 2^n integer numbers by Achilles heel operation [2]:

$$Y(2^n) = \frac{1}{\sqrt{2^n}}\left(\left|0_1\right\rangle + \left|1_1\right\rangle\right)\left(\left|0_2\right\rangle + \left|1_2\right\rangle\right)...\left(\left|0_n\right\rangle + \left|1_n\right\rangle\right) \tag{2}$$

Unfortunately, the $Y(2^n)$ time function will be zero most of the time because the probability that the sum within one parenthesis is not zero is 50% therefore the probability that the whole product is non-zero during a given clock period is $1/2^n$. Thus even though we can do a NOT operation on 2^n bits in a parallel and instantaneous way, we will not be

able to observe the result within a reasonable time frame by using the representation shown in Eq. 1.

The complex representation given below is free from this deficiency.

3. Complex RTW representation and its basic properties

3.1. *Real vs imaginary representation of bit values*

Let the stochastic time functions $A_k \equiv A_k(t)$ and $B_k \equiv B_k(t)$ $(k = 1,...n)$ be independent, discrete-time RTW signals with ± 1 random values, each value with 50% probability. The k-th complex noise bit is defined as:

$$\left|0_k\right\rangle = A_k \quad \text{and} \quad \left|1_k\right\rangle = \mathrm{i}B_k \tag{3}$$

where the presence of the imaginary unit **i** in a term indicates that the given RTW is in the "Im" wire while the absence of **i** indicates that the given RTW is in the "Re" wire.

To do algebra on the set of noise-bits, the rules of operations on complex numbers are applied. For example, the product state is built up by the complex product of the noise-bits where the rules of multiplying complex numbers are applied. Thus, a gate producing a product has 4 input wires and 2 output wires. Therefore, the product state $\left|x_1 x_2 ... x_n\right\rangle$ of n noise-bits will also be a complex number with the absolute value of 1. Typical noise-gates will perform operations so that the absolute value of the output complex signal remains 1.

For example, the NOT operation will swap the real and imaginary parts and Hadamard operation will produce:

$$\mathrm{H}_k\left|0_k\right\rangle = \frac{1}{\sqrt{2}}\left(\left|0_k\right\rangle + \left|1_k\right\rangle\right) = \frac{1}{\sqrt{2}}(A_k + \mathrm{i}B_k) \tag{4}$$

$$\mathrm{H}_k\left|1_k\right\rangle = \frac{1}{\sqrt{2}}\left(\left|0_k\right\rangle - \left|1_k\right\rangle\right) = \frac{1}{\sqrt{2}}(A_k - \mathrm{i}B_k) \tag{5}$$

A particular advantage of this representation is that the Achilles heel operation results in the product of complex numbers with unit absolute values thus the absolute value of $Y(2^n)$ is always 1 and never zero:

$$Y(2^n) = \frac{1}{\sqrt{2^n}}\left(\left|0_1\right\rangle + \left|1_1\right\rangle\right)\left(\left|0_2\right\rangle + \left|1_2\right\rangle\right)...\left(\left|0_n\right\rangle + \left|1_n\right\rangle\right) = \\ = \frac{\left(A_1 + \mathrm{i}B_1\right)}{\sqrt{2}}\frac{\left(A_2 + \mathrm{i}B_2\right)}{\sqrt{2}}...\frac{\left(A_n + \mathrm{i}B_n\right)}{\sqrt{2}} \tag{7}$$

3.2. *Complex conjugate representation of bit values*

Let the stochastic time functions $A_k \equiv A_k(t)$ and $B_k \equiv B_k(t)$ ($k = 1,...n$) be independent, discrete-time RTW signals with ± 1 random values, each value with 50% probability. The k-th complex noise bit is defined as:

$$\left|0_k\right\rangle = \frac{A_k + iB_k}{\sqrt{2}} \quad \text{and} \quad \left|1_k\right\rangle = \frac{A_k - \mathrm{i}B_k}{\sqrt{2}} \tag{8}$$

Similarly to the other complex representation shown above, to do algebra on the set of noise-bits, the rules of operations on complex numbers are applied. Therefore, the product state $\left|x_1 x_2 ... x_n\right\rangle$ of n noise-bits will also be a complex number with unity absolute value. Typical noise-gates will perform operations so that the absolute value of the output complex signal remains 1.

For example, the NOT operation is a simple complex conjugation. Similarly to the other complex representation above, Achilles heel operation results in the product of numbers with unit absolute values (these numbers are real) thus the absolute value of $Y(2^n)$ is always 1 and never zero:

$$Y(2^n) = \frac{1}{\sqrt{2^n}}\left(\left|0_1\right\rangle + \left|1_1\right\rangle\right)\left(\left|0_2\right\rangle + \left|1_2\right\rangle\right)...\left(\left|0_n\right\rangle + \left|1_n\right\rangle\right) = A_1 A_2 ... A_n \tag{9}$$

4. Ghost states and operations on products and superpositions

As we have mentioned above, many operations with large parallelism are possible on noise-bit product strings and their superposition by using simple algebraic manipulations such us multiplying the state with a polynomial expression of noise-bits, see in Section 3. However, the most powerful trick to do large-parallelism operations on noise-bits, including interacting noise-bits, and measurements, is based on "ghost" bits. Let us suppose that we have the superposition $Y(2^n)$ of 2^n product strings:

$$Y(2^n) = \sum_{k=1}^{2^n} c_k \left| x_{1k} x_{2k} ... x_{nk} \right\rangle = \sum_{k=1}^{2^n} c_k X_{1k} X_{2k} ... X_{nk} \tag{10}$$

We can generate another superposition $Y^*(2^n)$ that has exactly the same logic values but one or more of noise-bits are replaced by their "ghost" version, which are alternative RTWs that are independent of the original $2n$ RTWs of the original scheme. *Note, the star* $*$ *does not stand for complex conjugation but it indicates one or more ghost bits in the superposition, or if a star* $*$ *is used with a noise bit, it indicates a ghost bit*. That means, if we want to create the ghost bits of m noise-bits, we will need $2m$ extra independent RTW sources.

To illustrate the advantages of "ghost" bits, which are clearly due to the classical physical information properties of noise-based logic, let us see a few important examples. First let us generate $Y^*(2^n)$ with a ghost bit at the r-th position. Then the system states and RTWs are as follows:

$$Y(2^n) = \sum_{k=1}^{2^n} c_k \left| x_{1k} x_{2k} ... x_{nk} \right\rangle = \sum_{k=1}^{2^n} c_k X_{1k} ... X_{rk} ... X_{nk} \tag{11}$$

$$Y^*(2^n) = \sum_{k=1}^{2^n} c_k \left| x_{1k} x_{2k} ... x_{nk} \right\rangle = \sum_{k=1}^{2^n} c_k X_{1k} ... X^*_{rk} ... X_{nk} \quad , \tag{12}$$

where X^*_{rk} is the ghost bit. Thus we have two alternate systems, which represent the same logic superpositions, but the r-th noise bit values in them are represented by different RTWs. Some important applications are shown below. Without the restriction of generality, suppose the "real vs imaginary representation" of bit values.

i) Measurement of the value of a given bit in an arbitrary product string without knowing the values of the other bits. The product is:

$$Y(n) = \left| x_1 x_2 ... x_r ... x_n \right\rangle = X_1 X_2 ... X_r ... X_n \tag{13}$$

The task is to measure the value of the r-th bit. We generate the product with the same bit values but with ghost bit at the r-th position.

$$Y^*(n) = \left| x_1 x_2 ... x_r ... x_n \right\rangle = X_1 X_2 ... X^*_r ... X_n \tag{14}$$

Then we apply the following procedure:

$$\overline{Y}(n) Y^*(n) = \overline{X}_1 X_1 \overline{X}_2 X_2 ... \overline{X}_r X^*_r ... \overline{X}_n X_n = \overline{X}_r X^*_r \tag{15}$$

where the upper bar means complex conjugation and we used the $\overline{X}X = \left|X\right|^2 = 1$ relation.

Thus the $\overline{Y}(n)Y^*(n)$ product is one of these two possibilities:

if $x_r = 0$ then $\overline{Y}(n)Y^*(n) = A_r\, A_r^*$, or if $x_r = 1$, then $\overline{Y}(n)Y^*(n) = B_r\, B_r^*$ (16)

The $A_r\, A_r^*$ and $B_r\, B_r^*$ RTWs must be generated and compared with the $\overline{Y}(n)Y^*(n)$ RTW, see the relevant string verification method [7]. Observing just 83 clock steps, the probability is less than 10^{-25} [7] that we do not have the decision because the RTWs look identical during this time period. Thus, the measurement time and hardware does not depend on the length of the product when means the measurement requires $O(2^0)$ hardware and time complexity.

ii) Measure the weights of having $|0\rangle$ and $|1\rangle$ values at the *r*-th noise-bit in the superposition. The system states can be written as:

$$Y(2^n) = \alpha_r|0_r\rangle + \beta_r|1_r\rangle = \alpha_r A_r + \beta_r \mathrm{i} B_r \quad , \tag{17}$$

$$Y^*(2^n) = \alpha_r|0_r\rangle + \beta_r|1_r\rangle = \alpha_r A_r^* + \beta_r \mathrm{i} B_r^* \quad , \tag{18}$$

where α_r and β_r are noise timefunctions related to the zero and 1 values of the *r*-th noise-bit. In the case of a single product string, one of them is zero and the other one is the product of RTWs (thus it is itself an RTW) representing the values of the rest of the bits in the string. In the case of a superposition of product strings, α_r and β_r will also be superpositions and in large systems, a Gaussian noise properties can be expected.

Utilizing the facts that $\left(A_r\right)^2 = \left(A_r^*\right)^2 = \left(B_r\right)^2 = \left(B_r^*\right)^2 = 1$, and that A_r , A_r^*, B_r and B_r^* and known (reference) RTWs, this linear system of equation can easily be solved when $A_r\, B_r \neq A_r^* B_r^*$:

$$\alpha_r = \frac{Y(2^n)B_r - Y^*(2^n)B_r^*}{A_r\, B_r - A_r^* B_r^*} \qquad \beta_r = \mathrm{i}\,\frac{Y^*(2^n)A_r^* - Y(2^n)A_r}{A_r\, B_r - A_r^* B_r^*} \tag{19}$$

The weights of the $|0_r\rangle$ and $|1_r\rangle$ bit values are given by the time averages $\left\langle |\alpha_r|^2 \right\rangle_t$ and $\left\langle |\beta_r|^2 \right\rangle_t$, respectively.

Note, when $A_r\, B_r = A_r^* B_r^*$, the denominators in Equation 19 are zero thus that clock period must be omitted or we can use more parallel ghost operations due their low

hardware complexity and use that ghost amplitude, which satisfies the inequality. If we sue m parallel ghost bits, the probability that we still have a problem is $1/2^m$ thus, it disappears fast with increasing m.

iii) Single noise-bit operations in a large superposition of product strings. The procedure is as follows.

a) Follow the protocol described at point ii) until α_r and β_r are determined via Equation 19. Then multiply α_r and β_r by the relevant RTWs generated by the given logic gate acting on $|0_r\rangle$ and $|1_r\rangle$. In this way, the system's output state $Y_{out}(2^n)$ is synthesized where the r-th noise-bit values are replaced by the output of the logic gate depending on the actual values of the r-th bit in the given product string.

Example 1: To have the NOT gate acting on the r-th bit in all superposition terms, the following sum is needed:

$$Y_{out}(2^n) = \mathrm{NOT}_r\left[Y(2^n)\right] = \alpha_r|1_r\rangle + \beta_r|0_r\rangle = i\alpha_r B_r + \beta_r A_r \tag{20}$$

Note, by making the sum in Equation 20, we applied the NOT gate on the r-th bit in 2^n independent product strings. The hardware and time complexity we used go that is negligible and it is independent of n, thus it is $\mathrm{O}(2^0)$ complexity.

Example 2: To have the Hadamard gate acting on the r-th bit in all superposition terms, the following sum is needed:

$$Y_{out}(2^n) = \mathrm{H}_r\left[Y(2^n)\right] = \alpha_r \frac{|0_r\rangle + |1_r\rangle}{\sqrt{2}} + \beta_r \frac{|0_r\rangle - |1_r\rangle}{\sqrt{2}} = \alpha_r \frac{A_r + \mathrm{i}B_r}{\sqrt{2}} + \beta_r \frac{A_r - \mathrm{i}B_r}{\sqrt{2}} \tag{21}$$

By making the sum in Equation 21, we applied the Hadamard gate on the r-th bit in 2^n independent product strings. The hardware and time complexity we used go that is negligible and it is independent of n, thus it is $\mathrm{O}(2^0)$ complexity.

iv) Two noise-bit operations in a large superposition of product strings. The same concept as in *iii*) however now we must set up and solve the linear equations for both noise-bits which requires only two new RTWs to construct three pairs of ghost bit values and four independent linear equations. Considering the p-th and the r-th bits, we suppose that the two new RTWs are A_p^* and A_r^*. Thus the three pairs of ghost bit values can be expressed as

$$\begin{array}{l} A_p^*, B_p \text{ and } A_r, B_r \\ A_p, B_p \text{ and } A_r^*, B_r \\ A_p^*, B_p \text{ and } A_r^*, B_r \end{array} \tag{22}$$

And the interacting noise-bits are

$$\begin{aligned} Y(2^n) &= \alpha_{pr}\left|0_p 0_r\right\rangle + \beta_{pr}\left|0_p 1_r\right\rangle + \gamma_{pr}\left|1_p 0_r\right\rangle + \delta_{pr}\left|1_p 1_r\right\rangle = \\ &= \alpha_{pr} A_p A_r + \mathrm{i}\beta_{pr} A_p B_r + \mathrm{i}\gamma_{pr} B_p A_r - \delta_{pr} B_p B_r \end{aligned} \tag{23}$$

$$\begin{cases} Y^*\left(2^n\right) = \alpha_{pr} A_p^* A_r + i\beta_{pr} A_p^* B_r + i\gamma_{pr} B_p A_r - \delta_{pr} B_p B_r \\ Y^{**}\left(2^n\right) = \alpha_{pr} A_p A_r^* + i\beta_{pr} A_p B_r + i\gamma_{pr} B_p A_r^* - \delta_{pr} B_p B_r \\ Y^{***}\left(2^n\right) = \alpha_{pr} A_p^* A_r^* + i\beta_{pr} A_p^* B_r + i\gamma_{pr} B_p A_r^* - \delta_{pr} B_p B_r \end{cases} \tag{24}$$

After solving the (23-24) system of equations for α_{pr}, β_{pr}, γ_{pr} and δ_{pr}, we can obtain

$$\begin{cases} \alpha_{pr} = \dfrac{\left(Y - Y^*\right) - \left(Y^{**} - Y^{***}\right)}{\left(A_p - A_p^*\right)\left(A_r - A_r^*\right)} \\ \beta_{pr} = \dfrac{A_r^*\left(Y^* - Y\right) + A_r\left(Y^{**} - Y^{***}\right)}{iB_r\left(A_p - A_p^*\right)\left(A_r - A_r^*\right)} \\ \gamma_{pr} = \dfrac{A_p^*\left(Y^{**} - Y\right) + A_p\left(Y^* - Y^{***}\right)}{iB_p\left(A_p - A_p^*\right)\left(A_r - A_r^*\right)} \\ \delta_{pr} = \dfrac{A_r^*\left(A_p Y^* - A_p^* Y\right) + A_r\left(A_p^* Y^{**} - A_p Y^{***}\right)}{B_p B_r\left(A_p - A_p^*\right)\left(A_r - A_r^*\right)} \end{cases} \tag{25}$$

In accordance with Eq. 23, multiplying α_{pr}, β_{pr}, γ_{pr} and δ_{pr} by the relevant RTWs will synthesize the original superposition. In this way, the system's output state $Y_{out}(2^n)$ is synthesized where the *p*-th and *r*-th noise-bit values are replaced by the output of the logic gate depending on the values of the *p*-th and *r*-th bit in the given product string.

Example: To have the CNOT gate acting on the r-th bit with the p-th bit as control bit in all the superposition terms, the following sum is needed:

$$Y_{out}(2^n) = \mathrm{CNOT}_{pr}\left[Y(2^n)\right] = \alpha_{pr} A_p A_r + \mathrm{i}\beta_{pr} A_p B_r - \gamma_{pr} B_p B_r - \delta_{pr} B_p A_r \qquad (26)$$

By making the sum in Eq. 26, we applied the CNOT gate on the p-th and r-th bits in 2^n independent product strings. The hardware and time complexity we used go that is negligible and it is independent of n, thus it is $\mathrm{O}(2^0)$ complexity.

Whenever the zero-denominator problem emerges, it can be removed in the same ways as it is shown after Eq. 19.

5. Hardware complexity of two-bits operations

It is obvious from Section 3 above that by utilizing ghost states, a CNOT operation can be done on an exponentially large superposition with a polynomial hardware and time complexity. However, if the task is to execute k CNOT operations sequentially on different bit pairs in the product, when the results of all the sequential operations is extracted, we will need to introduce 2^k ghost RTWs. It is an open question, if this disadvantageous exponential hardware complexity need persists even for the situation when we do not need to know the results of all the sequential operations.

6. Conclusion

In this paper we made a further step toward the goal of emulating quantum computing algorithms by classical noise-based logic. We showed a way to represent the required product states and their superpositions and how to do the required basic single bit and two-bit operations with a high parallelism that is characteristics of idealistic quantum computers.

Reaching the ultimate goal would indicate that to efficiently run the algorithms developed for quantum computers, a classical noise-based logic hardware, including high-quality random number generators based on classical statistical physics (e.g. thermal noise), is enough.

Acknowledgments

This work has partially been supported by the National Natural Science Foundation of China under grant 61002035.

References

1. L.B. Kish, "Noise-based logic: Binary, multi-valued, or fuzzy, with optional superposition of logic states", *Physics Letters A* **373** (2009) 911–918.
2. L.B. Kish, S. Khatri, S. Sethuraman, "Noise-based logic hyperspace with the superposition of 2^N states in a single wire", *Physics Lett. A* **373** (2009) 1928–1934.

3. S.M. Bezrukov, L.B. Kish, "Deterministic multivalued logic scheme for information processing and routing in the brain", *Physics Lett. A* **373** (2009) 2338-2342.
4. L.B. Kish, S. Khatri, F. Peper, "Instantaneous noise-based logic", *Fluctuation and Noise Lett.* **9** (2010) 323-330.
5. F. Peper, L.B. Kish, "Instantaneous, non-squeezed, noise-based logic", *Fluctuation and Noise Lett.* **10** (June 2011), open access: http://www.worldscinet.com/fnl/00/0001/open-access/S0219477511000521.pdf
6. Z. Gingl, S. Khatri, L.B. Kish, "Towards brain-inspired computing", *Fluctuation and Noise Lett.* **9** (2010) 403–412.
7. L.B. Kish, S. Khatri, T. Horvath, "Computation using Noise-based Logic: Efficient String Verification over a Slow Communication Channel", *European Journal of Physics B*. **79** (2011) 85-90.
8. H. Wen, L.B. Kish, A. Klappenecker, F. Peper, "New noise-based logic representations to avoid some problems with time complexity", *Fluctuation and Noise Lett.*, **11**(2012), 1250003. http://arxiv.org/abs/1111.3859.
9. H. Wen, L.B. Kish, " Noise based logic: why noise? A comparative study of the necessity of randomness out of orthogonality", *Fluctuation and Noise Lett.*, vol. 11, no. 4, Dec, 2012, In press. http://arxiv.org/abs/1204.2545.