International Journal of Computer Applications (0975 - 8887)
Volume 125 - No.1, September 2015

Noise Tolerant Stochastic Logic Gate Circuits Synthesis using Genetic Algorithms

I. Neri
NiPS Laboratory, Dipartimento di Fisica e Geologia
Università degli studi di Perugia
I-06010 Perugia, Italy

F. Hartmann
Technische Physik and Wilhelm Conrad Röntgen Research
Center for Complex Material Systems, Physikalisches Institut
Universität Würzburg
Am Hubland, D-97074 Würzburg, Germany

ABSTRACT

In this paper we propose a method for synthesis of combinational networks using non conventional logic gates. The logic components considered are Stochastic Logic Gates (SLGs) able to change their logic functionality by means of a single control parameter and the environmental level of noise. SLGs are able to adapt their computed logic function depending on the environmental conditions. Circuits composed of SLGs are thus sensitive to changes in the environment which alter the computed logic function. We propose a solution for the synthesis of SLGs combinational networks able to produce a network operating fault tolerant in different environmental conditions, i.e. different levels of noise. Given a description of the problem, in form of a truth table, the synthesis of the network is performed by means of genetic algorithms. The proposed solution is tested with a half-adder and compared to the optimal solution found with an exhaustive search.

General Terms

optimization, circuits, noise

Keywords

stochastic logic gate, fault tolerant, genetic algorithm, non conventional computing

1. INTRODUCTION

In Logic Stochastic Resonance (LSR), bistable or multistable nonlinear dynamical system can function as a logic gate or memory device by exploiting the constructive interplay of noise and nonlinearity[1, 2]. Similarly to the well-known Stochastic Resonance (SR) concept [3, 4], the dynamical system enables to fulfil correct logical operations only for a non-vanishing noise floor and if the noise level is in an optimum range. A particular application of LSR are Stochastic Logic Gates. The most interesting characteristic of SLGs is the ability to perform correct logic functions in a noisy environment and the morphing of the logic function by tuning the potential landscape of the device[1] or the noise[2]. While the first can be chosen already in the design phase or changed before any logic operation, the latter depends strictly on the logic gate operation environment which in turn may change over time, e.g. due to local heating effects, and thus will modify the logic functionality of the gate.

In this work we present a method, based on evolutionary strategies, to synthesize a combinational logic network composed by SLGs fault tolerant to the environmental noise. Similar approaches have been used for evolving 100% functional circuits [5, 6, 7, 8] with focus on fault tolerance [9, 10, 11, 12].

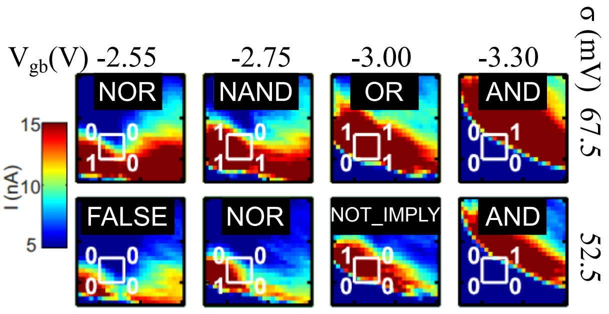


Fig. 1: Colour plots of the output current (logic output) versus Vgl and Vgr (logic inputs) for different values of noise (σ) and control parameter (Vgb). The corners of the white rectangles show the position of different side gate input-voltage configurations where Vgl and Vgr take either the high or the low level. High and low are defined to be $0V$ and $0.6V$ respectively. The output current levels are defined to be high if $I \geq 15nA$ and to be low if $I < 15nA$ [13].

2. EVOLVABLE HARDWARE

A particular example of SLGs is the one presented in Ref. [13] and in the simulated experiment presented here, we are considering this particular gate as a building block for the combinational network. In such SLGs the logical function performed by the gate does not simply depend on the noise level (σ) but also on a second control parameter (Vgb). Considering the combination of the two a large set of logic functionalities can be performed as depicted in Figure 1. In fact the device enables the realization of logic AND, OR, NAND and NOR gates, among others. Using this kind of SLGs, the control parameter is a property that can be set in the fabrication/production phase while the noise may vary with environmental conditions.

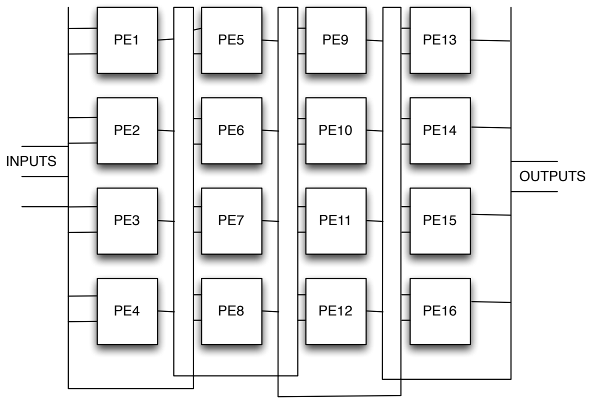


Fig. 2: Combinational network composed by reconfigurable processing elements (PEs). In this case the maximum number of PEs in the network is 16.

We envision an optimal scenario in which a network is completely fault tolerant even when the noise changes. To achieve this result it is necessary to design the logic network to be resilient to the noise parameter. For conventional logic gates there are several techniques to synthesize a combinational network in order to solve a specific problem, e.g. the Karnaugh map[14]. Considering non-conventional logic gates plus noise, it is almost impossible to obtain an explicit solution for the problem. Still, optimization techniques can be used to find a near-optimal solution to the problem.

For practical purposes it is useful to define an environment in which a combinational network made by SLGs is evaluated and designed. The logic architecture is evaluated considering a structure comparable to a virtual reconfigurable circuit (VRC) as used on FPGAs. The VRC is composed by an arbitrary number of processing elements (PEs). The general structure of the VRC is presented in Figure 2.

Each PE consists of two input selectors determining the input of the PE and thus the connection among PEs. A selector determines the logic function computed by the PE. Figure 3 represents the internal structure of a generic PE.

Notice that in the presented structure the inputs to the PEs of the first column are the problem inputs while the ones of the second column can access the problem inputs and the output of the first column, and so on. The outputs of the logic circuit can be any arbitrary set of PEs outputs. In order to use the optimization software to synthesize networks composed of SLGs presented above, instead of using the classical logic function (e.g. AND, OR, and so on), we define the following logic blocks:

—SLG_255;

—SLG_275;

—SLG_300;

—SLG_330;

corresponding to the SLGs for different values of the control parameter Vgb, i.e. the columns of Figure 1. The suffixes 255, 275, 300 and 330 refer to the voltage of the control parameter ($-2.55V$, $-2.75V$, $-3.00V$ and $-3.30V$ respectively). Each logic block may compute a different logic operation for different level of noise, corresponding to the rows of Figure 1. The logic outputs of the SLGs as function of inputs and noise levels are reported in Table 1.

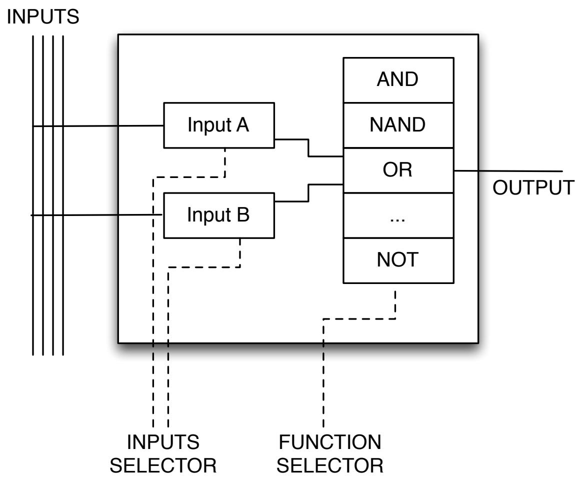


Fig. 3: Internal structure of a processing element. The processing element has a selector for the logical function to compute and two selectors for the inputs.

Table 1. : Logic outputs of the SLGs considered, as function of inputs and noise level.

σ	I_0	I_1	SLG_255	SLG_275	SLG_300	SLG_330
HIGH	0	0	1	1	0	0
HIGH	0	1	0	1	1	0
HIGH	1	0	0	1	1	0
HIGH	1	1	0	0	1	1
LOW	0	0	0	1	0	0
LOW	0	1	0	0	1	0
LOW	1	0	0	0	0	0
LOW	1	1	0	0	0	1

3. SYNTHESIS BY GENETIC ALGORITHMS

The synthesis of the combinational network is performed using optimization techniques inspired by nature, in particular we used Genetic Algorithms (GAs), in which individuals (i.e. candidate logic circuits) compete for reproduction. The individuals that fit better in the environment survive, reproduce, and thus transmit their genetic signature to the next generation. Eventually the chromosome of two individuals can be merged and mutated with some probability. In particular the GA evolution can be summarized in the following steps[15]:

(1) Read the problem definition.
(2) Generate an initial population of N individuals. Each individual defines a candidate solution for the problem encoding the structure of the solution in its chromosome.
(3) Evaluate the fitness of each individual in relation to the fitness function and the problem definition.
(4) Select candidate individuals for reproduction based on their fitness.
(5) Reproduction performing crossover operation and mutation operation.
(6) Generated new individuals form the next generation.
(7) Iterate from step 3.

The evolution can be halted once a particular condition is reached (e.g. maximum number of iteration reached or individual with target fitness found). In the following sections specific definitions of the algorithm for synthesis of logic circuit made by SLGs are given.

3.1 Problem definition

The target problem is defined by the truth table of the desired logic circuit. The number of the input and output of the truth table are mapped on the number of input and output of the VRC. In addition the number of rows and columns of the VRC is defined limiting the maximum number of PEs, and thus SLGs, to be used.

3.2 Genetic representation

Each individual is a realization of a VRC that defines the chromosome of the individual. Each PE represents a gene, defining the interconnection among PEs and the logic function performed.

3.3 Initial population

The initial population consists of a set of N randomly generated individuals, selecting with uniform distribution the logic function and the topological structure of the network. Generally a large population size will result in a faster convergence, measured in number of generations. This will incur in a larger computational cost. The right selection of the population size optimizes the overall computational time.

3.4 Fitness evaluation

The selection of the fitness function is a critical point in GA optimization. A good fitness function is meant to map the objective to a fitness value to be associated to each individual. The objective of the problem is to find a topology of SLGs able to satisfy the truth table considering to operate the logic network at two levels of noise. Moreover solutions that use fewer gates have to be preferred, reducing the complexity of the system and its energy consumption. In particular the fitness function is defined as:

$$f = \frac{\#correct\_outputs}{2 \times \#total\_outputs} - w * \#used\_gates$$

where $\#correct\_outputs$ is the number of correct outputs computed for both level of noise, and $\#total\_outputs$ is the number of outputs defined by the truth table. Finally $\#used\_gates$ is the number of PEs involved in the computation and w is a weight defining the importance of this parameter, in the presented case $w = 0.001$. The fitness function saturates to 1 when all the outputs are correct and no SLG is used.

3.5 Selection

The selection procedure is responsible for selecting the individual to be used to generate the next generation. The selected individuals can then be modified by specific genetic operations. In the presented work we used two different selection mechanism:

—Rank Selection

—Roulette Wheel Selection

For both selection methods the chances that individuals with higher fitness are selected are enhanced. In particular in the Rank Selection method the individuals are sorted by fitness and the selection probability is proportional to the rank of the individual in the sorted list[16]. In the Roulette Wheel Selection the probability of selection is proportional to the fitness of the individual giving a lesser selection pressure. Notice that a very high pressure may lead to a convergence to a sub-optimal solution [17]. In our case both selection methods have been used but lead to no substantial differences.

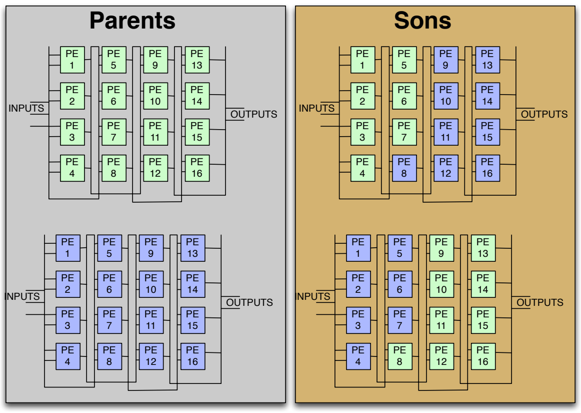


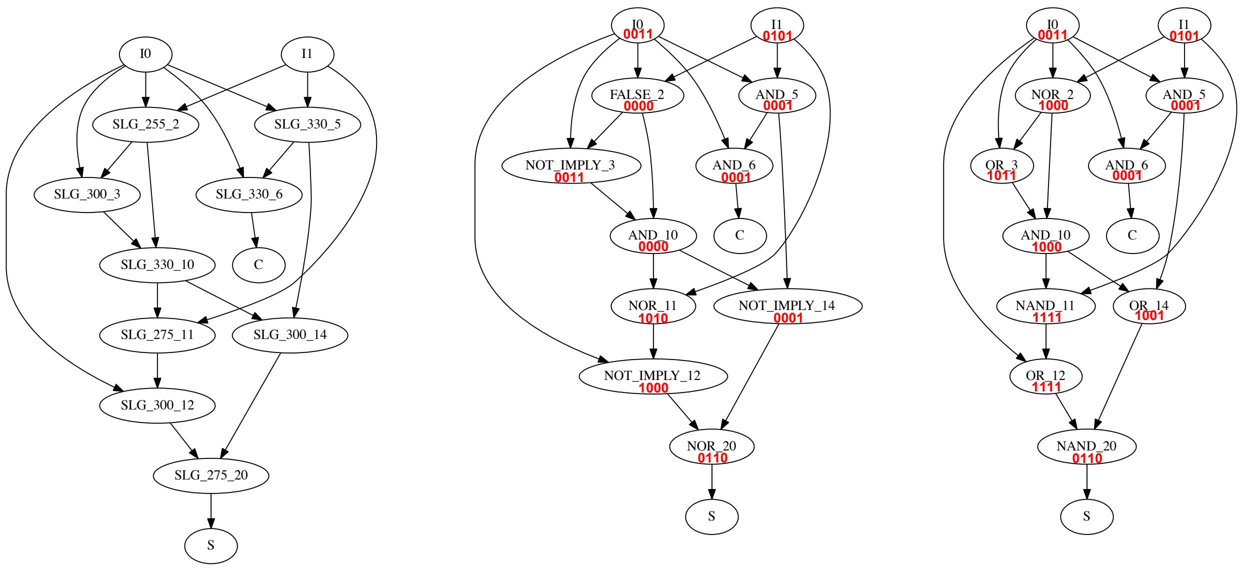
Fig. 4: Single-point crossover procedure. The VRC of the new individuals are composed by part of the PEs of one parent and part from the other parent.

3.6 Genetic operations

Reproduction of new individuals is carried out applying two genetic operations to the selected parents. The first operation is the crossover in which two parents are selected and eventually, with a given probability of 0.95, the two chromosome are mixed. The implemented crossover is a single point crossover consisting in dividing the chromosome in two parts and assign to a new individual the first part of the chromosome of one parent and the second part of the chromosome of the second parent. Vice-versa for the other new individuals[16, 18]. The chromosome is divided at PE level as illustrated in Figure 4.
The second genetic operation is the mutation. Mutation consists in randomly generate a new value for a specific property of a gene. In particular mutation operations permit to explore a larger space of parameter and eventually reintroducing a gene value lost during the selection procedure[16]. The mutation operation is usually performed with a low probability, in the presented case the probability of mutation is set to 0.15. The mutation operation occurs inside the PE eventually changing one of the input of the PE or its logic function. Moreover the mutation can occur at output level selecting as output of the problem the output of a different PE, or eventually directly one input.

4. SYNTHESIS AND SIMULATION RESULTS

We will consider evolving a circuit made by SLGs able to correctly compute the half-adder logic operation. The truth table of the half-adder is reported in Table 2.
The desired output logic circuit is thus a 2-inputs and 2-outputs circuit. While using traditional logic gates the solution to the problem is trivial (i.e. a XOR gate and a AND gate for the sum and carry respectively). No simple solutions are easy to find using the SLGs defined above, especially considering the requirement to work with both level of noise.
For the optimization procedure the maximum number of PEs to be used was set to 25, arranged in a matrix of 5×5. The population size was set to 30 individuals.



(a) Synthesized logic network.

(b) Logic network functionality with low noise.

(c) Logic network functionality with high noise.

Fig. 5: Synthesized fault tolerant half-adder using SLGs by genetic algorithms.

Table 2. : Truth table of the half-adder. I_0 and I_1 are the inputs while S and C are the sum and carry output bits respectively.

I_0	I_1	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

After 3000 generations the algorithm converges to a correct solution using 9 SLGs, represented in Figure 5a. The suffix at the end of the SLG name reefers to the position occupied in the VRC, and will be used to identify the gate in the following as #XX.
The two logic functions computed for the two levels of noise are shown in Figure 5b and Figure 5c for the lower and higher level of noise respectively. The logic function is presented along with the computed output in respect to the relative inputs. Analysing the particular case shows that two SLG_330 gates (#5 and #6) were used in cascade. Table 1 shows that SLG_330 acts as an AND gate for both levels of noise and thus the two AND gates connected in cascade are redundant and one of them can be removed reducing the number of needed gates to 8. Another peculiar feature of the proposed solution is the use of the gate SLG_255 (#2). For the lower level of noise it does not compute a logic operation and outputs are always zero (see Figure 5b). However for higher level of noise it acts as a NOR gate and becomes useful for the computation (see Figure 5c). In particular it is interesting to notice that part of the network is not useful for the computation in the low noise scenario while it is active in the high noise scenario or vice-versa. For instance in the low noise case (Figure 5b) the path up to the AND gate (#10) is useless, in fact it produces always 0 as output. However this port become important in the high noise level computing an output useful for the OR gate (#14), while the gates #11 and #12 become useless since their output is a constant 1.
As it happens with optimization techniques, the solution may be sub-optimal, and may correspond to a local minimum of the fitness function. To identify the optimal solution, considering the same set of SLGs, an exhaustive search have been performed. The optimal solution, consisting of 5 SLGs gates for the sum bit, and one SLG gate for the carry bit. The optimal solutions is shown in Figure 6.

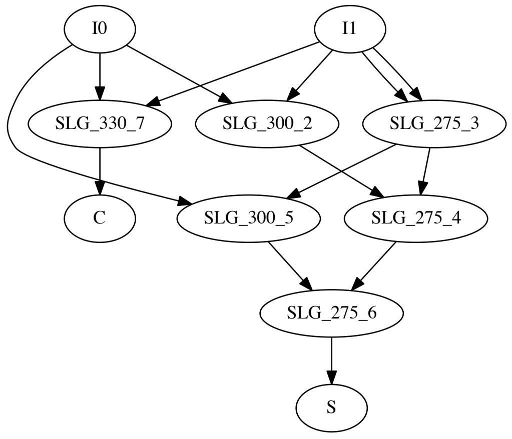


Fig. 6: Optimal solution for the half-adder using SLGs. Overall 6 logic elements are needed to realize a complete fault tolerant half-adder.

As expected, the optimal solution differs from the one found with the optimization algorithm, however for larger problems the use of exhaustive search is not feasible since the complexity of the problem grows exponentially with the number of gates involved. In fact even with today's high-powered computer the use of an exhaustive search to find an optimal solution can be prohibitively in terms of time even for a small problem. GAs are usually able to find a near-optimal solution exploring only a small subset of possible solution, reducing the computational time required.
The ability of a combinational circuit to be more robust to the noise takes its toll, usually in complexity or number of logic gate used. In fact, as mentioned, the last result found uses 6 gates instead of 2 in the traditional XOR/AND configuration. However, the trade-off between the number of gates, complexity and error has to be evaluated depending on the specific requirement of the application, energy constrains, and required reliance to noise. In the presented approach the desired trade-off can be selected *a-priori* setting the weights for each desired constrain in the fitness function.

5. CONCLUSIONS

In this work we proposed a genetic algorithm to create logic architectures with complete fault tolerance to the environmental noise by using stochastic logic gates. As test bench we considered a half-adder circuit. The found solution enables computation in a varying noise environment. The optimization is performed in order to maximize the functionality of the network and minimize the error and the energy consumption, reducing the number of gates used. The proposed solution can be used to synthesize logic architecture, using as building blocks non conventional logic gates, where classical synthesize techniques fail.

6. ACKNOWLEDGEMENTS

This work was supported by the European Commission under Grant Agreement No. 318287, LANDAUER and Grant Agreement No. 611004, ICT-Energy.

7. REFERENCES

[1] K Murali, Sudeshna Sinha, William L Ditto, and Adi R Bulsara. Reliable logic circuit elements that exploit nonlinearity in the presence of a noise floor. *Physical review letters*, 102(10):104101, 2009.

[2] F Hartmann, A Forchel, I Neri, L Gammaitoni, and L Worschech. Nanowatt logic stochastic resonance in branched resonant tunneling diodes. *Applied Physics Letters*, 98(3):032110, 2011.

[3] Kurt Wiesenfeld and Fernan Jaramillo. Minireview of stochastic resonance. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 8(3):539–548, 1998.

[4] Luca Gammaitoni, Peter Hänggi, Peter Jung, and Fabio Marchesoni. Stochastic resonance. *Reviews of modern physics*, 70(1):223, 1998.

[5] Julian F Miller, Peter Thomson, and Terence Fogarty. Designing electronic circuits using evolutionary algorithms. arithmetic circuits: A case study, 1997.

[6] Carlos A Coello, Alan D Christiansen, and Arturo Hernández Aguirre. Automated design of combinational logic circuits using genetic algorithms. In *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*, pages 335–338, 1997.

[7] Ahmed T Soliman and Hazem M Abbas. Combinational circuit design using evolutionary algorithms. In *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on*, volume 1, pages 251–254. IEEE, 2003.

[8] CK Vijayakumari, P Mythili, Rekha K James, and CV Anil Kumar. Genetic algorithm based design of combinational logic circuits using universal logic modules. *Procedia Computer Science*, 46:1246–1253, 2015.

[9] Didier Keymeulen, Adrian Stoica, Ricardo Zebulum, Yili Jin, and Vu Duong. Fault-tolerant approaches based on evolvable hardware and using a reconfigurable electronic devices. In *Integrated Reliability Workshop Final Report, 2000 IEEE International*, pages 32–39. IEEE, 2000.

[10] P Nirmal Kumar, S Anandhi, and J Perinbam. Evolving virtual reconfigurable circuit for a fault tolerant system. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 1555–1561. IEEE, 2007.

[11] Kyung-Joong Kim and Sung-Bae Cho. Automated synthesis of multiple analog circuits using evolutionary computation for redundancy-based fault-tolerance. *Applied Soft Computing*, 12(4):1309–1321, 2012.

[12] Hui-Cong Wu. Fault tolerant circuit design using evolutionary algorithms. *Journal of Computers*, 9(1):95–100, 2014.

[13] P. Pfeffer, F. Hartmann, S. Höfling, M. Kamp, and L. Worschech. Logical stochastic resonance with a coulomb-coupled quantum-dot rectifier. *Phys. Rev. Applied*, 4:014011, Jul 2015.

[14] M. Karnaugh. The map method for synthesis of combinational logic circuits. *American Institute of Electrical Engineers, Part I: Communication and Electronics, Transactions of the*, 72(5):593–599, Nov 1953.

[15] Thomas Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[16] Kit-Sang Tang, Kim-Fung Man, Sam Kwong, and Qun He. Genetic algorithms and their applications. *Signal Processing Magazine, IEEE*, 13(6):22–37, 1996.

[17] Pinaki Mazumder and Elizabeth M Rudnick. *Genetic algorithms for VLSI design, layout & test automation*. Prentice Hall PTR, 1999.

[18] David E Golberg. Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, 1989, 1989.