A Novel PSO Based Approach for Solving CMOL Cell Assignment Problem

Prateek Shrivastava and Khemraj Deshmukh

[1]Prateek Shrivastava, Electronics & Telecommunication/SSGI/ CSVTU, Bhilai India
[1]shrivastava.prateek7@gmail.com
[2]Khemraj Deshmukh, Electronics & Instrumentation/ SSGI/ CSVTU, Bhilai India
[2]khemraj.deshmukh@gmail.com

ABSTRACT

The genetic algorithm works with the concept of chromosomes having gene where each gene act as a block of one solution. This is totally based on the solution which is followed by crossover and then mutation and finally reaches to fitness. The best fitness will be considered as a result and implemented in the practical area. Due to some drawbacks and problems exist in the genetic algorithm implemented, scientists moved to the other algorithm technique which is apparently based on the flock of birds moving to the target. This effectively overcome the shortcomings of GA and provides better fitness solutions to implement in the circuit.

Keywords: Genetic Algorithm, CMOL Cell

1. INTRODUCTION

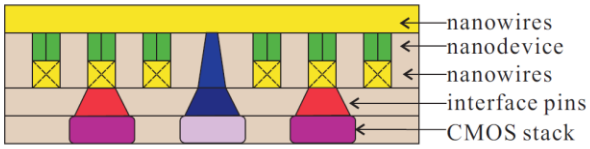
In recent years, nanoelectronics has made tremendous progress, with advances in novel Nano devices, Nano-circuits, Nano-crossbar arrays [1] manufactured by Nano imprint lithography[2] CMOS/Nano co-design architectures[3] and applications[4][5]. According to K. K. Likharev and D. V. Strukov discussed in Introduction to Molecular Electronics "Although a nanowire crossbar array (with two-terminal nanodevices) does not have the functionality of FET-based circuits, it has the potential for incredible density and low fabrication costs".

Likharev and his colleagues have developed the concept of CMOL (Cmos / nanowire /Molecular hybrid) as a likely implementation technology for nanoelectronic devices. Examples include memory, FPGA, and neuromorphic CrossNets [6].Cell assignment is one of the important steps in CMOL based circuit design.

One of the prior works on CMOL was assigning cells by hand. Another CMOL work[7] grouped CMOL cells into 4 × 4 tiles, where the CMOL cell assignment can be performed easily within each tile due to its small size, and the high level placement and route were performed at the tile level (but not the CMOL cell level) which is similar to traditional FPGA tools.

In Defect Tolerant CMOL Cell Assignment via Satisfiability given by William N. N. Hung, Changjian Gao, Xiaoyu Song

and Dan Hammerstrom, Instead of working with tile level abstraction, Hung[8] presented a technique that directly works on the CMOL cell level.



Consider the above figure; the nanowires are on top of CMOS circuits, with interface pins connecting CMOS metals and nanowires. The pins (blue) connecting with the upper-layer nanowires could break the lower-layer nanowires to relax the requirements for fabrication and increase interface yield.

The nanodevices are sandwiched between the two levels of perpendicular nano-imprinted nanowires. This unique structure solves the problems of addressing much denser nanodevices with sparser CMOS components. Each nanodevice is accessed by the two perpendicular nanowires which connect to the nanodevice. The nanowires are, in turn, connected by pins to the CMOS circuits. With N nanowires and pins, we could address O(N^2) nanodevices.

2. BACKGROUND

CMOL was originally developed by Likharev and his colleagues in Springer, Berlin, 2005. The nanodevice in CMOL can be any two terminal nanodevices, e.g. a binary "latching switch" based on molecules with two metastable internal states. The figure in above, shows the basic CMOL circuit with the interface between CMOS and nanowires. The pins connect the CMOS upper-level metals and the nanowires. The nanodevices are sandwiched between the two levels of perpendicular nano-imprinted nanowires.

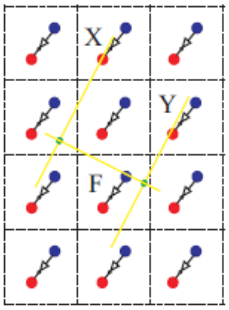
This unique structure solves the problems of addressing much denser nanodevices with sparser CMOS components. Each nanodevice is accessed by the two perpendicular nanowires which connect to the

nanodevice. The nanowires are, in turn, connected by pins to the CMOS circuits. With N nanowires and pins, we could address O(N^2) nanodevices.

Strukov and Likharev proposed the CMOL FPGA idea to fully explore the regularity of the CMOL architecture. Because the nanodevices are non-volatile switches, the CMOL FPGA could program those nanodevices and route the signals from CMOS to the nanowires and nanodevices, and back to CMOS again. Logic functions are created by a combination of CMOS inverters and diode-like nanodevices. To further explore architectural regularity, they proposed a cell-like CMOS structure, as shown in below figure. In each cell, there is a CMOS. Nanowires aligned with one direction receive signals from the outputs of the CMOS inverters. Those nanowires are OR'ed together with nanowires aligned with another (orthogonal) direction according to the nanodevice configurations. The OR'ed signal goes to the inverter's input, which is on the CMOS level. This OR-NOT logic is the preferred implementation of the CMOL FPGA.

For example, X, Y,F are three signals connected with the three grayed cells' output pins. With the illustrated nanowire connections (yellow lines) and "ON" nanodevices (green dots), the logic expression is

$$F = \overline{X + Y}.$$



When Strukov and Likharev first presented the CMOL FPGA, they performed the cell assignment task manually for several simple, regular-structured boolean circuits. They also presented a reconfigurable architecture for CMOL FPGA that grouped CMOL cells (e.g. 4×4 cells) to form tiles, which can be treated similar to traditional FPGA's clusters, and can utilize existing cluster-based FPGA CAD tools. However, that work also did not solve the CMOL cell assignment problem for the general case; because it was restricted to the tile abstraction and it relied on sufficient cell connectivity radius and the insertion of additional inverters for routing purposes.

In Defect Tolerant CMOL Cell Assignment via Satisfiability, the authors solved the CMOL cell assignment problem via satisfiability and extended it as a reconfiguration tool for various CMOL defects. However, satisfiability in general works for small to medium sized problems, but it does not scale nicely with the size of the problem. Therefore, we reinvestigate this problem as an optimization problem, with the hope of minimizing the area with reasonable CPU time.

In CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices. Nanotechnology,2005, Strukov and Likharev first presented the CMOL FPGA, they performed the cell assignment task manually for several simple, regular-structured boolean circuits. They also presented a reconfigurable architecture [7] for CMOL FPGA, that grouped CMOL cells (e.g. 4×4 cells) to

form tiles, which can be treated similar to traditional FPGA's clusters, and can utilize existing cluster-based FPGA CAD tools. However, that work also did not solve the CMOL cell assignment problem for the general case; because it was restricted to the tile abstraction and it relied on sufficient cell connectivity radius and the insertion of additional inverters for routing purposes.

According to above paper mentioned, there are periodic breaks in the nanowire fabric, such that each input/output nanowire has a fixed length based on these breaks. Hence each CMOL cell can only be connected to a limited number of neighbouring CMOL cells. The set of CMOL cells that can be connected to the input of a particular cell X is called the input **connectivity domain** of X. Similarly, the output connectivity domain refers to the set of cells that can be connected to the output of X.

Given any boolean combinational circuit, we can easily transform the circuit into a net list of NOR gates, using any library-based technology mapping from any logic synthesis tool or the technique outlined in [8]. Mathematically, we can view the NOR gate net list as a directed acyclic graph (DAG) G:

$$G = (V,E)$$

where V is a set of vertices (each vertex correspond to a NOR gate in the net list), and $E = V \times V$ is a set of directed edges. Given a collection of CMOL cells Ψ. We need to find a mapping from the vertices (NOR gates) to the CMOL cells,

$$p : V \rightarrow \Psi$$

2.1 Genotype

The genotype is the genetic constitution of a chromosome and for our CMOL cell assignment problem, it is a solution which represent the mapping result. The genotype structure is (C1, C2, C3, . . . , CN), where $N = ||\Psi||$ is the number of cells in the CMOL cell array. Each Ci can be either an index to an NOR gate or -1. The NOR gate index is in the range [0,M −1], where M is the total number of NOR gates in the circuit. The value -1 indicate the cell has not been occupied by any NOR gate. Each position 1 …N in the chromosome corresponds to a CMOL cell in the array.

2.2 Fitness Function

In genetic algorithm, fitness function is used to evaluate the adaptive ability. This function provides a measurement standard for choosing the best population to the next generation. The CMOL cell assignment problem requires the wire lengths of connected NOR gates to be less than a constant R, the radius of the connectivity domain.

2.3 Genetic Operators

1) Crossover: Crossover is one of the main genetic operators in the GA. It takes two parent solutions to reproduce their children. After the selection process, the population is enriched with better individuals. In the process of selection, we adopted Roulette Algorithm to choose the top best parents to crossover based on their fitness and survival probability.

2) Mutation: Mutation occurs after crossover. It is used to prevent the algorithm from being trapped in local minima. Mutation is viewed as an operator which recovers lost genetic pattern as well as randomly disturbs genetic information.

3. PROPOSED METHODOLOGY

3.1 Particle Swarm Optimization and Its Variants

A new method for optimization of continuous nonlinear functions was introduced **by** J. Kennedy and R. Eberhart**.**

Particle swarm optimization can be used to solve many of the same kinds of problems as genetic algorithms **(GAS)**. This optimization technique does not suffer, however, from some of GA's difficulties; interaction in the group enhances rather than detracts from progress toward the solution. Further, a particle swarm system has memory, which the genetic algorithm does not have.

Change in genetic populations results in destruction of previous knowledge of the problem, except when elitism is employed, in which case usually one or a small number of individuals retain their "identities." In particle swarm optimization, individuals who fly past optima are tugged to return toward them; knowledge of good solutions is retained by all particles.

3.2 The Particle Swarm Optimization Concept

A concept for the optimization of nonlinear functions using particle swarm methodology is introduced.

Particle swarm optimization has roots in two main component methodologies. Perhaps more obvious are its ties to artificial life (A-life) in general, and to bird flocking, fish schooling, and swarming theory in particular. It is also related, however, to evolutionary computation, and has ties to both genetic algorithms and evolutionary programming.

Particle swarm optimization **as** developed by the authors comprises a very simple concept, and paradigms can be implemented in a few lines of computer code. It requires only primitive mathematical operators, and is computationally inexpensive in terms of both memory requirements and speed.

Particle swarm optimization is **an** extremely wimple algorithm that **seems** to be effective for optimizing a wide range of functions. We view it **as** a mid-level form of A-life or biologically derived algorithm, occupying the space in nature between evolutionary search, which requires eons, and neural processing, which occurs on the order of milliseconds. Conceptually, it **seems** to lie somewhere between genetic algorithms and evolutionary programming.

Particle swarm optimization is similar to a genetic algorithm in that the system is initialized with a population of random solutions. It is unlike a genetic

algorithm, however, in that each potential solution **is** also assigned a randomized velocity, and the potential solutions, called particles, are then "flown" through hyperspace.

Each particle keeps track of its coordinates in hyperspace which are associated with the **best** solution (fitness) it **has** achieved so far. (The value of that fitness is also stored.) This value is called pbest. Another "best" value is also tracked. The "global" version of the particle **swarm** optimizer keeps track of the overall best value, and its location, obtained thus far by any particle in the population; this is called gbest.

The particle swarm optimization concept consists of, at each time step, changing the velocity (accelerating) each particle toward its pbest and gbest (global version).Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward pbest and gbest.

4. EXPECTED OUTCOME

The whole analysis and implementation of approach is depending on the following outcome:

1. Increasing Area Utilization.
2. Reducing CPU time.
3. Reducing the number of CMOL cells.
4. Reducing the Area.

ACKNOWLEDGMENT

Useful discussions with respected Khemraj Deshmukh and Anil kumar sahu are gratefully acknowledged. The work has been supported by the IJREST.

REFERENCES

[1] Philip J. Kuekes, Duncan R. Stewart, and R. Stanley Williams. The crossbar latch: logic value storage, restoration, and inversion in crossbarcircuits. Journal of Applied Physics, 97:034301–1–5, 2005.

[2] D. J. Resnick. Imprint lithography for integrated circuit fabrication. Journal of Vacuum Science & Technology B: Microelectronics and Nanometre Structures, 21:2624, 2003.

[3] K. K. Likharev and D. V. Strukov. CMOL: devices, circuits, and architectures. In G Cuniberti and et al., editors, Introduction to Molecular Electronics, pages 447–477. Springer, Berlin, 2005.

[4] Dmitri B. Strukov and Konstantin K. Likharev. CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices. Nanotechnology, 16(6):888–900, 2005.

[5] Oˆ . Tu¨rel, J. H. Lee, X. Ma, and Konstantin K. Likharev. Architectures for nanoelectronic implementation of artificial neural networks: new results. Neurocomputing, 64:271–283, 2005.

[6] D. B. Strukov and K. K. Likharev. Prospects for terabit-scale nanoelectronic memories. Nanotechnology, 16:137–148, 2005.

[7] D. Strukov and K. Likharev. A reconfigurable architecture for hybrid CMOS/nanodevice circuits. In FPGA'06, pages 131–140, Monterey, California, USA, 2006.

[8] William N. N. Hung, Changjian Gao, Xiaoyu Song, and Dan Hammerstrom. Defect Tolerant CMOL Cell Assignment via Satisfiability. IEEE Sensors Journal, 8(6):823–830, June 2008.

