Cells Reconfiguration Around Defects in CMOS/Nanofabric Circuits Using Simulated Evolution Heuristic

Abdalrahman M. Arafeh and Sadiq M. Sait
Department of Electrical and Computer Engineering
University of British Columbia, Vancouver, Canada
Department of Computer Engineering and
Center for Communications and IT Research, Research Institute
King Fahd University of Petroleum & Minerals
Dhahran-31261, Saudi Arabia
email: arafeh@ece.ubc.ca, sadiq@kfupm.edu.sa

Abstract— Recent advances in nanoscale components assembly have led to the invention of low-power and high-density nanofabrics, which can be integrated with conventional CMOS transistors. CMOS/nanofabric hybrid circuits combine the flexibility and high fabrication yield advantages of CMOS technology with ultra fast nanometer-scale devices. CMOL is a novel architecture which consists of a nanofabric overlay on top of a CMOS stack. CMOL can be configured to implement NOR-based logic circuits by programming nanodevices placed between the nanofabric's overlapping nanowires. Defects rate in nanofabric-based circuits is expected to be higher than that of conventional CMOS technology. Misassembly of nanodevices will lead to non-programmable crosspoints, while broken nanowires will result in unreachable circuit's components. In such cases, utilizing CMOS/nanofabric architectures requires robust reconfiguration-based defect-tolerance design automation tools that can circumvent defective components and insure circuits functionality.

In this work, we propose a heuristic-based nanofabric reconfiguration around defective nano-components in CMOL circuits. Simulated Evolution (SimE) is formulated to find circuits configurations that adhere to nanowires connectivity constraint and rely on non defective components. Circuits of various sizes from ISCAS'89 benchmarks were used to evaluate our proposed design. Results show that SimE yield successful reconfigurations in acceptable computation time when up to 50% of nanodevices are stuck-at-open and 70% of nanowires are broken.

Keywords— CMOL, Nanofabrics, Reconfiguration, Defects, Simulated Evolution, VLSI, Design Automation.

I. Introduction

A shift from CMOS-based integrated circuits to chemically assembled nanoelectronics has been under investigation in the past years. Conventional CMOS fabrication suffers from multiple physical limitation when gate length is scaled down to few-nm region [1]. Recently, many proposals focused on the so-called *bottom-up self-assembly* approach; where the smallest active devices of integrated circuits are not defined lithographically but assembled from components with reproducible size and structure. Those components are based on the advances in single electron devices [2], quantum dot cells [3], and reconfigurable switches [4].

Self-assembly process is based on synthesizing and connecting nano-scale components (e.g., wires or devices) through chemical process [5]. Two-dimensional nanofabrics are formed by the intersection of two sets of orthogonal parallel nanometer-sized wires. Nanowires are aligned to construct a reconfigurable array with nanometer spacing, where the formed crosspoints can be used as programmable nanodevices. Post-manufacturing configuration can render the assembled nanofabrics into functional circuits; the small active nanodevices can be programmed to pass electrical signals and to implement logic functions. Nanoscale crossbar systems offer ultra-high density and can operate at THz frequencies [6], however, nanowires and nanodevics are likely to have many imperfections.

Circuits consisting of nanoscale molecular devices alone are hardly viable as they lack register structures and mechanisms for signal restoration. Many suggested architectures integrate two-dimensional nanofabrics with CMOS-based subsystem, which provides the needed signals latching and the additional necessary functionalities (e.g., high voltage gain). Following this approach, researchers proposed hybrid CMOS/nanofabric circuits with varying layouts and organizations; examples of which include Likharev et al. CMOL circuits [7] and Crossnets [8], and Goldstein et al. Nanofabrics [9]. Recent reviews of CMOS/nanofabric circuits can be found in [10].

Defects are major issue for devices with few atoms in diameter. The small cross-section and contact areas can render nanodevices fragile and defect prone. Although the common problem of nanofabric-based architectures is high defect density, yet it is still possible to utilize the defect-free nano-components. Nanofabric's faulty devices can be detected and located using testing and diagnosis techniques; where the nanofabric is tested in conjunction with an outside circuit or by configuring part of it to test its own resources based on Built-in Self-test (BIST) techniques. Tahoori et al. [11] surveyed different approaches for defect detection and diagnosis in nanofabrics computing. Defect-tolerant nanofabric-based circuits implementations can be achieved by utilizing the reconfigurable and defect-testable nanodevices and through employing effective design tools that perform post-manufacturing circuits *Reconfiguration*.

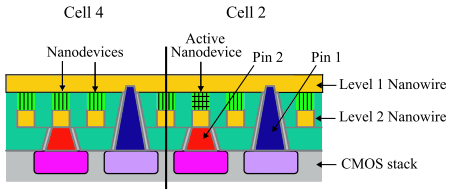
Nanofabrics reconfiguration can be achieved by either; reconfiguring circuits around defective components using an already available defect information stored in a defect map, or following a defect-unaware approach; where defect-free subsets within the original defective nanofabric are identified and then used for circuits implementation. For the defect-unaware approach; Huang et al [12] presented a defect-tolerance scheme which uses defect density information obtained from the manufacturing process to determine the expected size of functional (defect-free) crossbars. Whereas, Tahoori et al. [13] modelled the nanofabric as a bipartite graph where a greedy algorithm defines the defect-free maximum biclique. The algorithm looks for defect-free $k \times k$ crossbars within the original defective $n \times n$ nanofabric.

Finding successful reconfigurations for large nanofabrics is not guaranteed as the given problem is NP-complete [14]. The aforementioned proposed algorithms are only applicable for small nanofabric crossbars; greedy algorithms are expected to have degraded results and to consume considerable computation time in case of high defect rates. Furthermore, mapping large circuits with hundreds of gates and thousands of connections to subsets of defect-free nanofabric crossbars will further require an additional step to insure the global connectivity among the mapped subcircuits. In that sense, in this paper we investigate how a non-deterministic heuristic, namely, Simulated Evolution-SimE, can be formulated to utilize defect information, which are stored in defect maps, to find successful circuits reconfigurations in CMOL hybrid CMOS/nandevices architecture. Our purposed approach employ a global reconfiguration policy, where SimE iteratively improves a given circuit configuration until defective components are all avoided.

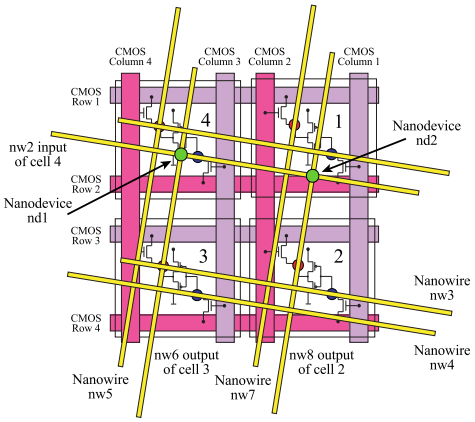
In the next section we give a review of CMOL hybrid CMOS/nanodevices architecture. Section III provides the problem formulation of nanofabrics reconfiguration. Design and characterization of Simulated Evolution non-deterministic heuristic are presented in Section IV. Section V highlights our experimental setup and results. Finally, we conclude with final remarks.

II. CMOL FPGA-like Circuits

CMOL cell-based, Field-Programmable Gate Array (FPGA)-like circuits integrate conventional inverter-based four-transistor MOSFET CMOS cells with uniform reconfigurable nanofabric [7]. Each CMOS cell (four cells are shown in Figure 1(b)) consists of an inverter and two pass transistors. Two-terminal nanodevices "latching switches", which have two metastable internal states and diode-like I-V curves, are assembled at each nanowires crosspoint. Likharev et al. predicted the density of nanodevices to be above 10^{12} to cm^2 for $F_{nano} = 3\ nm$, where F_{nano} is the nanowires half-pitch. This arrangement results in abundance of available nanodevices which can serve both inter-cells connectivity and wiring-logic. CMOS stack is connected with the nanofabric by metal pins that span to



(a)Schematic side view of two CMOL cells with two levels of nanowires. Only one nanodevice is activated to connect the output of Cell 2 with the input of Cell 4.



(b)Four CMOL cells and corresponding pins and nanowires. Only two nanodevices are shown; $nd1$ and $nd2$ which connect the outputs of Cell 2 and Cell 3 with the input of Cell 4.

Fig. 1. Low-level structure of CMOL circuit.

top and bottom nanowire levels as shown in Figure 1(a). Metal pins (i.e., Pin 1 and Pin 2) connect cell's input and output respectively with the nanofabric. Any two CMOS cells (e.g., cell '2' and cell '4') can be connected through a pin-nanowire-nanodevice-nanowire-pin connection.

Four inverter-based CMOS cells and their corresponding nanowires are shown in Figure 1(b). Cell '3' (i.e., lower left cell) is connected to cell '4' (i.e., upper left cell) through a combination of two nanowires (nw2 and nw6) and a nanodevice (nd1). Nanodevice nd1 is located at the intersection between nanowire nw6 (which is connected with output pin of cell '3') and nanowire nw2 (which is connected with input pin of cell '4'). When two or more nanodevice on cell '4' input nanowire (nw2) are activated (i.e., nd1 and nd2) the output of the cell will be equivalent to a NOR gate whose inputs are cell 2 and cell 3. The wired-OR logic is implemented through nanowires and nanodevice and the NOT logic is performed by the cell's CMOS inverter.

CMOL nanofabric is rotated by angle $\alpha = \arcsin(F_{nano}/\beta F_{CMOS})$ related to the CMOS pins which are arranged into a square array with side of $2\beta F_{CMOS}$, where F_{CMOS} is CMOS half-pitch, and β is a factor greater than 1. Each CMOS cell has an area of $A = (2\beta F_{CMOS})^2$. Like other nanofabric crossbars, CMOL's nanowires break at repeated intervals of $L = 2\beta^2 F_{CMOS}^2$ confining CMOL cells connectivity to only $M = a^2 - 2$ other cells located within a square-shaped *Connectivity Domain* shown in Figure 2, where a is a positive integer number

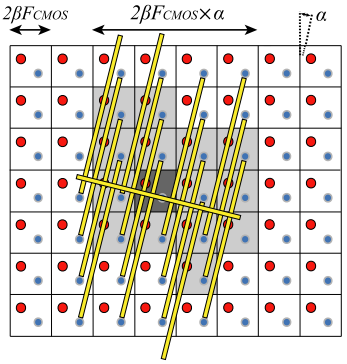


Fig. 2. CMOL FPGA-like Architecture: Connectivity Domain. Cells in light gray comprise the connectively domain of the cell in dark gray. $a = 4$ and $M = a^2 - 2 = 14$.

which indicates the connectivity domain radius and represents the constraint on CMOL cells connectivity. In Case of $a = 4$, the output pins of cells painted in light-gray in Figure 2 can be connected to the input pin of the specified dark-gray cell.

Each nanodevice in CMOL nanofabric is uniquely addressed via the appropriate pins pair (i.e., one input pin and one output pin in two different cells). CMOL can be physically configured using an address decoder which selects two CMOS columns and two CMOS rows (i.e., selects a pair of CMOL cells), then inverters are turned off and pass transistors are used for setting the binary state of each molecular device by relaying appropriate configuration voltages. When configuration is done the nanodevices are either set ON (low-resistance) state or OFF (high-resistance). By turning the programmable diodes "ON" or "OFF"; nanowires, nanodevices and CMOS inverters can implement basic wired NORs with multiple fan-ins/fan-outs. The advantage of CMOL architecture is that gates with high fan-in and fan-out may be readily formed as well by turning on the corresponding latching switches (i.e., nanodevices). Further, the architecture is inherently defect-tolerant, since it has $M \approx a^2 >> 1$ nanodevices per CMOS cell, and few of them are required for either logic or interconnect functionality.

A. CMOL Circuits Implementation

Implementing circuits in CMOL (i.e., mapping NOR-based logic gates to CMOL hybrid fabric) requires for the connected gates to be placed in CMOL cells, which are proximate to each others and for the connections to use non-defective components. Conventionally, CMOL cells mapping is performed through two sequential steps: 1) defect-unaware cells *Placement*, where logic gates are assigned to CMOL cells obeying the connectivity domain constraint, and failure to do so require the insertion of intermediate buffers to connect distant cells. 2) Cells *Reconfiguration* around defective components, as the arrangement laid out by placement may depend on defective nanofabric components. In this step, cells are reshuffled to overcome defects.

Recently, several proposals had been introduced for defect-unaware cell *Placement* in CMOL circuits. Likharev et al. utilized existing FPGA CAD tools to perform placement and routing on 4×4 tile-based version of CMOL [15], [16]. Hung et al. [17] encoded CMOL cell placement as a Satisfiability problem at cells level. Previous attempts to use sub-optimal search heuristics for CMOL placement were reported in [18], [19], [20]. Genetic Algorithm (GA) [18] were used with two dimensional block PMX crossover operator and mutation. A more elaborate work was reported in [19]; where Memetic computing approach was used by implementing a hybrid of Genetic Algorithm and Simulated Annealing (SA) local-based search heuristic. Xia et al. [20] extended the work on Memetic approach by integrating self-learning operators using Lagrangian Multipliers (LRMA). Other attempts to solve CMOL cell placement problem were based on Simulated Evolution [21] and Tabu Search [22]. The authors have shown that an arrangement of cells, which completely obey the connectivity constraint, can be found. Compared with best published techniques, Tabu Search and SimE have respectively shown over 90% and 82% reduction in average CPU processing time.

Reconfiguration around defective CMOL nanodevices was reported by Likharev et al. [7]. They developed a reconfiguration algorithm assuming only one type of defect (stuck-at-open). The algorithm preforms a number of sequential attempts to move each gate from a cell with bad input or/and output connections (i.e., connections which use defective nanodevices) to a new cell. Each gate may be swapped with another one, provided that all connections of the swapped gates can be realized under CMOL connectivity constraint and are not defective. The authors showed case studies by reconfiguring KoggeStone adder and full crossbar around randomly distributed defects. Other attempt to reconfigure CMOL circuits were reported using SAT solvers [17]. Few clustered defects (around single source) were introduced in small and medium sized CMOL circuits. A center of mass computation is performed to focus on a limited reconfiguration region where defects are formulated as satisfiability constraints.

The aforementioned reconfiguration techniques are expected to be sufficient for small or medium size circuits. Reconfiguration becomes more complex if CMOL has high defect rates and wider range of defects or when circuits' gates have many fan-in or fan-out connections. In such cases, resorting to SAT or sequential-attempts algorithms will be insufficient and circuits reconfiguration may fail.

III. **Problem Formulation**

The implementation of combinational logic using CMOL involves the assignment of logic gates (i.e., NORs or Inverters) to cells that are connected by defect-free programmable nanodevices. There are tens of thousands of nanodevices (i.e., possible connections) in a CMOL circuit, however, those connections will not be used simultaneously, but a small subset of them is sufficient to map a particular

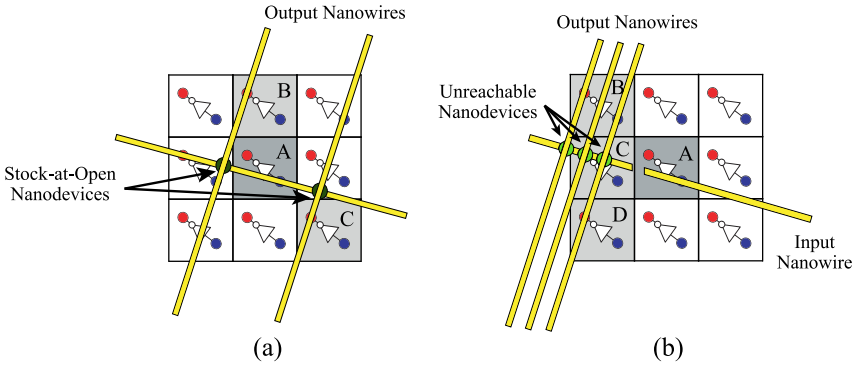


Fig. 3. Defects in CMOL circuits: (a) Type 1: Stuck-at-Open nanodevices defect (b) Type 2: broken nanowire defect. Cells shown in light gray are not reachable by the cell in dark gray.

circuit. The extra nanodevices are mainly intended for better reconfigurability. Circuits in CMOL can be configured in field by first performing a defect-unaware cells placement [21], and then apply non-deterministic heuristics to reconfigure cells locations as to avoid defective components. The reconfiguration procedure need to be applied once per circuit and each CMOL circuit will have its own unique defect map which can be acquired by utilizing BIST techniques. The scope of this work is to investigate the second stage of CMOL cell's mapping (i.e., reconfiguration). We have considered two types of widely used defect models:

1. Stuck-at-open: The nanodevice which connects two perpendicular nanowires is stuck-at-open (i.e., not programmable). In this case, the connection between two given CMOL cells through this nanodevice is not feasible. However, each of those two CMOL cells can still be used and connected with other cells.
2. Broken Nanowire: An input or output nanowire of a given CMOL cell is broken into two segments. Thus, this CMOL cell may not be connected to all other cells within its input/output connectivity domains. The connectivity domains of the affected cell is significantly reduced.

Type 1 and type 2 defects are shown in Figure 3. In stuck-at-open defect; the output of cells B and C cannot be connected with the input of cell A as the nanodevices between them are not programable. In broken nanowire defect shown in Figure 3(b); the number of unreachable cells (e.g., cells B, C and D) from a given cell (e.g., cell A) could be larger as more connections are affected by the cut in the nanowire. In this type of defect, each unreachable nanodevice can be considered as if it's stuck-at-open. Similar to other nanofabric-based circuits; CMOL is susceptible to stuck-at-close defects. Even a small percentage of stuck-at-close defects can results in having most of CMOL cells connected with each others. Different approaches other than reconfiguration should be investigated to circumvent stuck-at-close defects in nanofabric-based circuits.

Based on the discussed defect models; CMOL reconfiguration problem can be stated as follows: for a set of gates $G = g_1, g_2, g_3, ..., g_m$ and a set of nets $\Gamma = \gamma_1, \gamma_2, \gamma_3, ..., \gamma_m$ where $\gamma_i = \{fan\text{-}in_i \ \& \ fan\text{-}out_i\}$ of g_i and given a set of slots or locations (i.e, cells) $L = L_1, L_2, L_3, ..., L_n$ where $m \leq n$, each gate g_i in location L_i which uses defective connections is reassigned to a new location L_j given that the connectivity domain is respected and defects are avoided. According to CMOL FPGA topology shown in Figure 2, if a particular gate is moved to another location it will use different set of nanodevices to connect with its fan-in and fan-out gates. Reconfiguration should not relocate two gates in which their connectivity is violated, but rather to only avoid using any defective components. Mathematically, reconfiguration constraint can be defined as follows; Given a gate and its net $(g_i, \gamma i)$ placed in location L_i, for any gate $g_k \subseteq G$ and g_k in the net γi the following equation should be satisfied.

$$\forall g_i \in G, \exists g_k \in \gamma_i : N(L_i, L_k) \neq 0 \tag{1}$$

Where $N(L_i, L_k)$ is the nanodevice connecting gate g_i in location L_i and gate g_k in location L_k. $N = 0$ means the nanodevice is defective. CMOL reconfiguration is intended to rearrange cells to honor the constraints in Equation 1, and meanwhile not violating the constraints of CMOL connectivity domain defined as follows:

$$\forall g_i, g_k \in G : (g_i \neq g_k) \Rightarrow (L_i \neq L_k) \tag{2a}$$
$$\forall g_i \in G, \exists g_k \in \gamma_i : L_k \in C(L_i) \tag{2b}$$

Where L_k is the location of g_k, $C(L_i)$ is the connectivity domain of cell in location L_i. Reconfiguration is highly dependent on defect rates and connectivity radius a.

A. Defect Maps

In nanowire crossbars, imprecision and non-determinism of the nanoscale assembly process may cause the programmable nanodevices to be defective. Different methods for simulating faults distribution has been reported in the literature [23]. In this work, two methods are used for stuck-at-open faults simulation. In the first approach, a uniform random distribution is used. For any given nanodevice, a random number p is generated, the nanodevice could be defective if p is less than a pre-defined defect rate q_{nano}. In the second approach, clustered faults are injected around multiple defect sources. Each cluster is generated

as follow; first a random location (x_0, y_0) is chosen, and then a probability mass function $pmf(x, y)$ is computed for each location using the Gaussian distribution:

$$pmf(x,y) = Ce^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}} \quad (3)$$

This probability mass function controls the injection of faults; where C is a constant that sets the density of the simulated faults, and σ is the standard deviation that controls the diameter of the defects cluster. For each nanodevice we generate a random number p between 0 and 1. A fault is injected if $p \leq pmf(x, y)$. In broken nanowires defect; a nanowire is cut if a randomly generated number p is less than wires cut rate q_{wire}, and the cut point is randomly specified. All unreachable nanodevices on the cut nanowire are then encoded as if they are stuck-at-open.

IV. Simulated Evolution

The Simulated Evolution algorithm is a general search strategy for solving a variety of combinatorial optimization problems. It is stochastic because the selection of which elements to be reallocated is done according to a stochastic rule. Already well located elements have a high probability to remain where they are. In SimE heuristic the movable elements have a goodness value (a number between 0 and 1). Those with goodness value close to 1 have a smaller possibility of leaving their current locations, while those with smaller goodness (i.e., ill suited components) have otherwise.

The structure of the SimE is shown in Algorithm 4. SimE assumes that there exists a solution ϕ of a set G of m (movable) elements or gates. The algorithm starts from an initial cells assignment $\phi_{initial}$, and then, following an evolutionary-based approach it seeks to reach a better assignments from one generation to the next by perturbing some ill-suited components (i.e, gates) and retaining the near-optimal ones. A cost function $Cost$ associates with each assignment of movable element g_i a cost C_i. The algorithm has one main loop consisting of three basic steps, Evaluation, Selection, and Allocation. The three steps are executed in sequence until the solution average goodness reaches a maximum value, or no noticeable improvement to the solution cost is observed after a number of iterations. The Evaluation step consists of evaluating the goodness $goodness_i$ of each element g_i of the solution ϕ. The goodness measure, a number expressible in the range [0, 1], can be defined as the fraction of two values related to the problem cost.

The second step of the SimE algorithm is Selection. Selection takes as input a bias value B, the solution ϕ together with the estimated goodness of each element. The selection operator has a non-deterministic nature, i.e., an individual with a high goodness (close to one) still has a non-zero probability of being assigned to the set of selected elements S. It is this property of non-determinism that gives SimE the hill climbing capability to escape local minimum. Allocation is the next SimE operator, it generates a new complete solution ϕ' with the elements of the selection set S mutated according to an allocation function. The goal of Allocation is to favor improvements over the previous generation [24]. Details of the main steps of SimE algorithm for CMOL nanofabric reconfiguration problem are given in the following text.

ALGORITHM *Simulated_Evolution*$(B, \Phi_{initial}, StoppingCond.)$
NOTATION
B= Bias Value. Φ= Complete placement.
g_i= Gate i. $goodness_i$= Goodness of g_i.
$ALLOCATE(g_i, \Phi')$=Function to allocate g_i in new solution Φ'
Begin
Repeat
EVALUATION:
ForEach $g_i \in \Phi$ evaluate $goodness_i$;
/Evaluates if elements use defective nanodevices*/
SELECTION:
ForEach $g_i \in \Phi$ **DO**
begin
IF $Random \leq 1 - goodness_i + B$
THEN
begin
$S = S \cup g_i$; Remove g_i from Φ
end
ALLOCATION:
ForEach $g_i \in S$ **DO**
begin
$ALLOCATE(g_i, \Phi')$
/* Reallocate elements as to avoid using */
/* defective nanodevices. */
end
Until *Stopping Condition is satisfied*
Return Best solution.
End (*Simulated_Evolution*)

Fig. 4. Structure of Simulated Evolution algorithm.

A. Solution Representation

Solutions are represented as an arrangement of logic cells in two dimensional layout surface. The layout size corresponds to the number of required CMOL cells to fit each benchmark circuit. The outer cells of the grid are reserved for the circuit's I/O ports. Reconfiguration process starts from a complete cells placement, in which all gates are placed within the connectivity domain of their nets (i.e., connectivity domain constraint is honored). Each gate is represented by a unique positive integer value.

B. Goodness Function

In Simulated Evolution, goodness function is used to evaluate individual gates in each generation, where unfit gates are selected and reassigned to other cells. The goodness function of each individual gate is defined as:

$$goodness_i = \frac{connect_i}{|\gamma_i|} \quad (4)$$

Where $connect_i$ represents the number of connections in set γ_i that do not use defective nanodevices (i.e., the connections that are defect free) and $|\gamma_i|$ is the number of fanin and fanout gates of gate g_i. The above equation assumes a minimization problem (or a maximization of goodness). When gate g_i's connections violate the constraint in Equation 1, the gate will have low goodness value.

C. Cost Function

The main objective of the reconfiguration is to tolerate defects and ensure that the circuit is working properly. The

overall cost of a solution is expressed as the total number of used defective nanodevices (i.e., the number of connections that violate Equation 1). Equation 5 shows the cost of each gate $g \in G$ as to be the number of defective components it uses to connect with its fan-in and fan-out cells. The overall circuit's cost is the summation of individual gates cost.

$$C_i = \sum_{j \in fan\text{-}out_i} u_{i,j} \tag{5a}$$

$$u_{i,j} = \begin{cases} 1 & \text{if } N(L_i, L_j) = 0 \\ 0 & \text{otherwise} \end{cases} \tag{5b}$$

D. Selection and Allocation

In the Selection function, a gate is selected for reallocation if its goodness score is less than a randomly generated number between 0 and 1. The higher the goodness value of the element, the higher is the chance of retaining its current location. While, lower the goodness value, the more likely the element will be perturbed and reallocated.

Allocation function intends to generate a new solution that is inherently better than the old one. The function is fully aware of the presence of defects. For each selected gate, the allocation function evaluates the cost of swapping the gate with another one in the grid based on the cost function in Equation 5. Then, the best swap is chosen. An additional constraint also applies for gates movements in circuit's reconfiguration; the reallocation of gates is constrained to the region defined by the intersection of the connectivity domains of the two cells under investigation and their fanin and fanout cells. This insure that reconfiguration do not invalidate the assignment made by placement phase (i.e., do not move cells in which some of the connections violate the constraints in Inequality 2(b)).

E. Defective Nets Resolving

When Reconfiguration stage terminates, there may still be some connections that use defective nanodevices or cut nanowires. In such case, the circuit will not be functional and defective connections need to be mitigated by rerouting them through additional buffers (i.e., different set of nanodevices). Intermediary pair of inverters are inserted between cells which have their connections faulty. Buffers insertion is performed by recalling Simulated Evolution with little modifications. Inverters are placed in empty locations (i.e., blank cells) so that two cells with defective interconnect become connected. The constraints highlighted in Equation 1 and Equation 2 should also apply for the inserted inverters. SimE allocation function is modified as to only permit the interchange of two inserted inverters or an inverter and a blank cell. The other already occupied CMOL cells are assumed to be fixed and no perturbation can involve any one of them.

V. Experimental Results

Given defect distributions discussed in Section 3; we have performed our experiments using two defect maps; a randomly generated map ($R1$) and a clustered map ($C1$), which have $C = 0.8$ and standard deviation $\sigma = \frac{2a}{3}$. Evaluation of the search heuristics efficiency and performance was done using two scenarios based on different values for the probability of nanodevices stuck-at-open defect q_{nano} (type 1 defect), and the probability of a broken nanowire defect q_{wire} (type 2 defect). Scenario (i) includes five experiments when q_{nano} ranges between 10% to 50%, while $q_{wire} = 20\%$, and similarly scenario (ii) is comprised of seven experiments when q_{wire} probability is between 10% and 70% and $q_{nano} = 20\%$.

Reconfiguration is applied to 19 circuits of different sizes from ISCAS'89 benchmarks suite. Circuits are mapped by SIS synthesis tool to a NOR netlist with maximum of five inputs. Details of ISCAS'89 circuits are shown in Table II; the numbers of *Cells* including *Gates*, *Inputs* and *Outputs* are given. The table also gives CMOL 2-D grids sizes which are used to implement ISCAS'89 benchmarks; *Area* (*Row* × *Column*) indicates the number of *Rows* and *Columns* and the number of *CMOL Cells* which a given benchmark uses. Defect maps $R1$ and $C1$ are generated for each grid size; benchmarks that need similar CMOL gird sizes are reconfigured using same defect maps.

Simulated Evolution is implemented using Java programming language and run on a machine with 1.5 GHz Intel Pentium M processor and 512MB memory. The Heuristic stops when solution's cost (i.e., number of defective nanodevices) becomes zero or when a predefined number of 4000 iterations is reached. The average value of results obtained from 20 successful reconfigurations for each circuit is reported, where each run uses different seeds for random numbers. Figure 5 shows the change of problem cost (Equation 5) per iteration for s1238 benchmark circuit; SimE heuristic evolves to better solutions without being too greed. In some iterations, SimE perform hill-climbing by making bad moves (i.e., accept solutions with higher cost) as that may lead to better solutions and help avoid local minimum solutions.

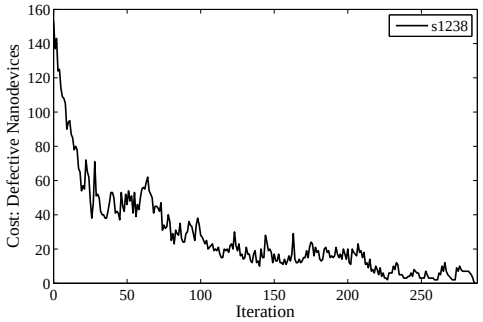


Fig. 5. Change of reconfiguration cost per iteration - s1238.blif.

A. Reconfiguration Results

To reconfigure circuits in CMOL, we have adhered to the original description of the connectivity domain shown in Figure 2 with connectivity radius $a = 18$. We have used a bias B between $[-0.06, 0.05]$, depending on the circuit's

defect rate. Negative bias was used for high defect rates, in order to reduce the number of selected gates for reallocation (particularly, in early iterations). This help in preventing the heuristic from keep making conflicting moves, which results in poor search space exploration. Whereas, positive bias values were used for low defect rates scenarios, as to explore the search space more vigorously by perturbing some well-suited gates. Our implementation is the first attempt to solve CMOL reconfiguration problem through non-deterministic heuristics. Results reported in this section were not compared with those based on Satisfiability [17] (discussed in Section 2-A), due to the limited nature of that implementation, which included only a small number of defects (less than 10% of the overall nanodevices).

Results for defect scenario (i) are given in the following tables; Table I reports the maximum and averaged CPU computation times (in seconds) needed to reconfigure all benchmark circuits for random and clustered defect maps when q_{nano} is between 10% and 40%. For this given range of stuck-at-open probabilities, SimE was successful in reconfiguring circuits around all defective components and no intermediary buffer were required. As defect probability rises the computation time increases. The maximum CPU time corresponds to the large benchmark circuits (e.g., s1238) and those that has many multiple fan-in NOR gates (e.g., s820, s832).

TABLE I

Reconfiguration CPU computation time for Scenario (i) and defect maps $R1$, and $C1$ ($q_{nano} = 10\% - 40\%$ - $q_{wire} = 20\%$).

Maps	q_{nano}	Max	Avg.
R1	10%	0.51	0.11
	20%	0.96	0.20
	30%	2.46	0.41
	40%	6.50	0.92
C1	10%	0.64	0.14
	20%	1.38	0.25
	30%	2.72	0.52
	40%	6.50	1.26

Max.: Maximum CPU computation time.
Avg.: Average CPU computation time.
q_{nano}: Probability of Stuck-at-Open nanodevices's defects.

Table II shows Simulated Evolution results for nanodevices stuck-at-open defect probability $q_{nano} = 50\%$. *Time* is CPU computation time in seconds and *Buffers* is the number of inserted buffers to resolve defective nets. For $q_{nano} = 50\%$ only s820 and s832 needed additional buffers to resolve defective nets. Those two circuits also needed more buffers when defects are clustered (e.g., s832 required 3 buffers in $R1$, and 6 in $C1$). Moreover, SimE's reconfiguration of clustered defects consumed more CPU time. For example, $C1$ average CPU time is 3.11 seconds, compared to 2.01 seconds for random map $R1$.

Reconfiguration results for defect scenario (ii) are shown in Table III. The reconfiguration of $s820$ and $s1238$ circuits is performed when up to 70% of the nanowires are cut and 20% of the nanodevices are stuck-at-open. SimE has found successful reconfigurations without the need for any additional buffers even when the probability of broken wires is as high as 70%.

In investigating the ratio of successful reconfigurations, we have run SimE for benchmark s1238 using 20 different clustered defect maps. All defect maps have $C = 0.8$, $\sigma = \frac{4a}{3}$, and cut rate $q_{wire} = 20\%$. The heuristic was run 40 times for each map; the overall successful reconfigurations rate when $q_{nano} = 50\%$ was 60% (i.e., 60% of the 40 run $\times$ 20 maps = 800 run). For defect maps with defect rate $q_{nano} < 50\%$ the overall successful reconfigurations was over 95%.

VI. Conclusion

In this paper, we have presented the formulation of a design automation algorithm (i.e., Simulated Evolution), as a foundation for a reconfiguration-based defect-tolerance scheme for the emerging hybrid CMOS/nanofabric architectures. A defect-free configurations were found through iterative improvement of the circuits defect-tolerance. Our findings showed that nanofabric-based architectures can still be utilized even though nanodevices suffer from high defect rate. Two different defect distributions were used to evaluate the performance of the applied heuristic. Obtained results showed that circuits can be reconfigured to become functional even if 50% of CMOL nanodevices are stuck-at-open or if up to 70% of the architecture's nanowires are broken.

VII. Acknowledgements

Authors acknowledge King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia, for all support.

References

[1] D.J. Frank, R.H. Dennard, E. Nowak, P.M. Solomon, Y. Taur, and Hon-Sum Philip Wong. Device scaling limits of Si MOSFETs and their application dependencies. *Proceedings of the IEEE*, 89(3):259 –288, mar 2001.

[2] Sergey Kubatkin, Andrey Danilov, Mattias Hjort, Jerome Cornil, Jean-Luc Bredas, Nicolai Stuhr-Hansen, Per Hedegard, and Thomas Bjornholm. Single-electron transistor of a single organic molecule with access to several redox states. *Nature*, 425(6959):698–701, 2003.

[3] P. Douglas Tougaw and Craig S. Lent. Logical devices implemented using quantum cellular automata. *Journal of Applied Physics*, 75(3):1818–1825, 1994.

[4] C. P. Collier, E. W. Wong, M. Belohradsky, F. M. Raymo, J. F. Stoddart, P. J. Kuekes, R. S. Williams, and J. R. Heath. Electronically configurable molecular-based logic gates. *Science*, 285(5426):391–394, 1999.

[5] V.V. Zhirnov and D. J C Herr. New frontiers: self-assembly and nanoelectronics. *Computer*, 34(1):34–43, 2001.

[6] Michael Butts and Andre DeHon. Molecular electronics: Devices, systems and tools for Gigagate, Gigabit chips. In *In ICCAD-2002*, pages 433–440, 2002.

[7] 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.

[8] O"zgr Trel, Jung Hoon Lee, Xiaolong Ma, and Konstantin K Likharev. Neuromorphic architectures for nanoelectronic circuits. *International Journal of Circuit Theory and Applications*, 32(5):277–302, 2004.

[9] S. Copen Goldstein and M. Budiu. NanoFabrics: spatial computing using molecular electronics. In *Computer Architecture, 2001. Proceedings. 28th Annual International Symposium on*, pages 178 –189, 2001.

TABLE II

ISCAS'89 benchmarks, the corresponding CMOL's grid sizes in ($Row \times Column$), and Simulated Evolution reconfiguration results for defect Scenario (i) and defect maps $R1$, and $C1$, when $q_{nano} = 50\%$ - $q_{wire} = 20\%$.

Circ.	Cells	Gates	Inputs	Outputs	Area ($Row \times Column$)	$R1$ - $q_{nano} = 50\%$		$C1$ - $q_{nano} = 50\%$	
						Time	Buffers	Time	Buffers
s27	19	8	7	4	$25(5 \times 5)$	0.03	0	0.03	0
s208	136	109	18	9	$169(13 \times 13)$	0.03	0	0.03	0
s298	122	85	17	20	$144(12 \times 12)$	0.06	0	0.19	0
s344	180	130	24	26	$196(14 \times 14)$	0.06	0	0.10	0
s349	184	134	24	26	$196(14 \times 14)$	0.10	0	0.10	0
s382	175	124	24	27	$196(14 \times 14)$	1.22	0	0.32	0
s386	164	138	13	13	$196(14 \times 14)$	3.78	0	3.10	0
s400	188	137	24	27	$196(14 \times 14)$	0.64	0	0.64	0
s420	299	248	34	17	$361(19 \times 19)$	0.32	0	0.26	0
s444	187	136	24	27	$196(14 \times 14)$	0.77	0	0.35	0
s510	304	266	25	13	$361(19 \times 19)$	1.09	0	2.34	0
s526	273	222	24	27	$324(18 \times 18)$	2.21	0	1.06	0
s641	302	206	54	42	$676(26 \times 26)$	0.61	0	1.22	0
s713	321	225	54	42	$676(26 \times 26)$	0.64	0	1.79	0
s820	447	400	23	24	$529(23 \times 23)$	8.06	3	12.13	4
s832	454	407	23	24	$529(23 \times 23)$	9.18	3	18.27	6
s838	606	507	66	33	$676(26 \times 26)$	1.31	0	1.79	0
s1196	675	613	31	31	$729(27 \times 27)$	3.04	0	6.24	0
s1238	724	662	31	31	$784(28 \times 28)$	4.99	0	9.12	0
Avg.	-	-	-	-	-	2.01	1	3.11	1

TABLE III

SimE results for s820 and s1238 circuit's reconfiguration around cut nanowires (Scenario ii), when $q_{nano} = 20\%$.

q_{wire}	$s820$				$s1238$			
	$R1$		$C1$		$R1$		$C1$	
	Time	Buffers	Time	Buffers	Time	Buffers	Time	Buffers
10%	0.29	0	0.29	0	0.38	0	0.45	0
20%	0.42	0	0.48	0	0.58	0	0.54	0
30%	1.15	0	1.12	0	0.83	0	0.86	0
40%	1.34	0	2.59	0	0.86	0	1.73	0
50%	2.56	0	10.11	0	2.21	0	1.95	0
60%	7.20	0	13.47	0	3.46	0	3.04	0
70%	13.60	0	26.14	0	4.77	0	5.54	0
Avg.	3.79	0	7.74	0	1.87	0	2.02	0

[10] A. DeHon and K.K. Likharev. Hybrid CMOS/nanoelectronic digital circuits: devices, architectures, and design automation. In *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, pages 375 – 382, nov. 2005.

[11] Mehdi B. Tahoori and Subhasish Mitra. Fault detection and diagnosis techniques for molecular computing. In *in NanoTech*, 2004.

[12] Jing Huang, M.B. Tahoori, and F. Lombardi. On the defect tolerance of nano-scale two-dimensional crossbars. In *Defect and Fault Tolerance in VLSI Systems, 2004. DFT 2004. Proceedings. 19th IEEE International Symposium on*, pages 96 – 104, oct. 2004.

[13] M. B. Tahoori. A mapping algorithm for defect-tolerance of reconfigurable nano-architectures. In *Proceedings of the 2005 IEEE/ACM International conference on Computer-aided design*, ICCAD '05, pages 668–672, Washington, DC, USA, 2005. IEEE Computer Society.

[14] Yadunandana Yellambalase and Minsu Choi. Cost-driven repair optimization of reconfigurable nanowire crossbar systems with clustered defects. *J. Syst. Archit.*, 54(8):729–741, August 2008.

[15] Dmitri B. Strukov and Konstantin K. Likharev. A reconfigurable architecture for hybrid CMOS/Nanodevice circuits. In *Proceedings of the 2006 ACM/SIGDA 14th international symposium on Field programmable gate arrays*, FPGA '06, pages 131–140, New York, NY, USA, 2006. ACM.

[16] Dmitri B. Strukov and Konstantin K. Likharev. CMOL FPGA circuits. In *In Proc. of Int. Conf. on Computer Design, CDES2006*, pages 213–219, 2006.

[17] William N.N. Hung, Changjian Gao, Xiaoyu Song, and D. Hammerstrom. Defect-tolerant CMOL cell assignment via satisfiability. *Sensors Journal, IEEE*, 8(6):823 –830, june 2008.

[18] Yinshui Xia, Zhufei Chu, William N.N. Hung, Lunyao Wang, and Xiaoyu Song. CMOL cell assignment by genetic algorithm. In *NEWCAS Conference (NEWCAS), 2010 8th IEEE International*, pages 25 –28, june 2010.

[19] Zhufei Chu, Yinshui Xia, William N.N. Hung, Lunyao Wang, and Xiaoyu Song. A memetic approach for nanoscale hybrid circuit cell mapping. In *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, pages 681 –688, sept. 2010.

[20] Y. Xia, Z. Chu, W. Hung, L. Wang, and X. Song. An integrated optimization approach for nano-hybrid circuit cell mapping. *Nanotechnology, IEEE Transactions on*, PP(99):1, 2011.

[21] Sadiq M. Sait and Abdalrahman M. Arafeh. Efficient CMOL nanoscale hybrid circuit cell assignment using Simulated Eevolution heuristic. In *Proceedings of the great lakes symposium on VLSI*, GLSVLSI '12, pages 21–26, New York, NY, USA, 2012. ACM.

[22] Sadiq M. Sait and Abdalrahman M. Arafeh. Cell assignment in hybrid CMOS/nanodevices architecture using Tabu Search. *Applied Intelligence*, pages 1–12, 2013.

[23] C.H. Stapper. Simulation of spatial fault distributions for integrated circuit yield estimations. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 8(12):1314 –1318, dec 1989.

[24] Sadiq M. Sait and Habib Youssef. *Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems*. IEEE Computer Society Press, California, December 1999.