Arab J Sci Eng (2015) 40:2515–2529
DOI 10.1007/s13369-015-1682-1

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

Reconfiguration-Based Defect-Tolerant Design Automation for Hybrid CMOS/Nanofabrics Circuits Using Evolutionary and Non-Deterministic Heuristics

Sadiq M. Sait[1,2] · **Abdalrahman M. Arafeh**[3]

Received: 9 December 2014 / Accepted: 23 April 2015 / Published online: 5 May 2015

Abstract Recently, a shift from CMOS lithography to nanoelectronics chemical assembly has been under investigation. Nanoscale components are assembled into arrays of low-power and high-density nanofabrics, which can be integrated with conventional CMOS systems. The inability to achieve inexpensive defect-free mass manufacturing of nanoelectronics is the largest impediment of their adoption. Limited nanowire lengths and defect-prone nanodevices pose significant challenges for design automation tools. In this work, we propose a design phase for cell mapping and reconfiguration in novel hybrid CMOS/nanoelectronics architecture called CMOL. *Reconfiguration* consists of finding new suitable physical location for each gate such that the circuit becomes defect free. The novelty of this work is to engineer non-deterministic iterative search heuristics, namely simulated evolution (SimE) and Tabu search (TS), to find cell assignment around defective nano-components. Circuits of various sizes from ISCAS'89 benchmarks are used to evaluate the designed heuristics. Results show that SimE and TS yield successful reconfigurations in reasonable computation time when nanodevice defect rate is as high as 50 % and nanowire cut rate up to 70 %.

Keywords Combinatorial optimization · CAD · CMOL · Nanofabrics · Reconfiguration · Defects · Simulated Evolution · Tabu Search · VLSI · Design automation

1 Introduction

Scaling down CMOS lithography process into nanometer region brings about a number of serious challenges. Among those challenges are fabrication plant costs and sub-micron physical limitations like short channel effects and doping fluctuations [1,2]. A shift from CMOS-based integrated circuits to chemically assembled nanoelectronics has been under investigation in the past years. The new nanoelectronics circuits take advantage of small, basic and active components with reproducible size and structure. Those components are based on the recent advances in single electron devices [3,4], quantum dot cells [5], reconfigurable switches [6] and negative differential resistors (NDR) [7].

Molecular-sized nanoelectronics are very small in which they need to be chemically self-assembled instead of being lithographically fabricated. Self-assembly process is based on synthesizing and connecting nanoscale components (e.g., wires or devices) through chemical processes [8]. A group of nanowires are self-aligned to construct an array (i.e., two-dimensional orthogonal grid) with nanometer-scale spacing. At each intersection of two wires, a configurable component (e.g., switch or device) is formed. Self-assembly and self-alignment result in simple orthogonal grid-shaped nanofabrics, where post-manufacturing field configuration can render them into functional circuits.

Abdalrahman M. Arafeh was at King Fahd University of Petroleum & Minerals when this work was done.

✉ Sadiq M. Sait
sadiq@kfupm.edu.sa

Abdalrahman M. Arafeh
arafeh@ece.ubc.ca

1 Department of Computer Engineering, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia

2 Center for Communications and IT Research, Research Institute, King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia

3 Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, Canada

Overcoming nanometer-scale lithography limitations with *bottom-up* components' self-assemble comes at a price. The small-size devices and the fundamental non-determinism of the chemical process give rise to high defect density. Defects are the major issue for devices with few atoms in diameter. The small cross sections and contact areas can render nanoelectronics fragile and defect prone. The order of defects in the nanofabrics surpass the highly controlled lithography-based CMOS manufacturing process.

Nanofabrics have a built-in defect tolerance provided by reconfigurable and defect-testable nanodevices along with alternative nanowire connections. The configurability and rich interconnects offer the choice between different nanowires and nanodevices to implement a particular circuit. Reconfiguring a circuit involves reprogramming nanodevices (set each of them to be "ON" or "OFF") based on new *Configuration Data*. Nanofabrics pose a compromise between simple and imprecise manufacturing process and complex design automation tools, which are needed to implement circuits by configuring nanofabrics around defective components.

Nanoelectronics can operate at THz frequencies [9] and consume less power as switching only involves few electrons. Moreover, their small size (i.e., few nanometers) offers huge density improvement over conventional CMOS technology, which entails multiple microscale transistors per logic cell. However, the absence of transistors in nanoelectronic systems constitutes functional restrictions such as lack of register structures and mechanisms for signal restoration. Many suggested systems use CMOS to buffer signals and provide the additional necessary functionalities (e.g., high voltage gain). Following this approach, many researchers proposed hybrid CMOS/nanodevices circuits with different layouts and organizations, examples of which include Likharev et al. CMOL circuits [10] and CrossNets [11] and Goldstein et al. Nanofabrics [12]. Recent reviews of CMOS/nanodevices circuits can be found in [13–16].

Nanowire interconnects are local in nature (i.e., they have limited length), which reduces the impact of a particular defect to only a small subset of the nanofabric. Nanofabrics can be tested in conjunction with an outside circuit or by configuring part of it to test its own resources. Once a defect map is generated, defect information can be used to avoid defective components when implementing a particular circuit. Tahoori et al. [17] surveyed different approaches for defect detection and diagnosis in nanofabrics computing. Different variations and enhancement to BIST have been proposed, among those are designs reported in [18–20].

Nanoelectronic fabrics are promising high-density and low-power circuits. The main impediment for the adaptation of nanoelectronics is their imprecise manufacturing process. This work aims to help speed up the adaptation of nanoelectronics through advances in design automation tools that can map cells to arrays with defects through circuit reconfiguration. Harnessing the power of the very small molecular devices can be done by employing effective design tools that map logic circuits to nanofabrics and perform post-fabrication reconfiguration around defects. The novelty of this work is to engineer iterative heuristics to find an assignment around defective nano-components. The main contributions of this papers are:

1. A design automation phase for circuit placement and reconfiguration in CMOL hybrid CMOS/nanodevice architecture via the engineering of iterative heuristics.
2. A formulation of CMOL reconfiguration problem and extensive experimentations using different defect maps.
3. Addressing the solution to overcome three major sources of defects (stuck-open, cut nanowires and defective CMOS cells) by iterative improvement of circuit's configuration.

The paper is organized as follows; in the next section, we give a review covering CMOL hybrid CMOS/nanodevices fabric and discuss previous solutions for CMOL cell placement and reconfiguration. Section 3 provides problem formulation of gates-to-nanofabric mapping and design automation flow. Engineering and implementation of non-deterministic heuristics namely simulated evolution (SimE) and Tabu search (TS) are presented in Sect. 4. Section 5 includes the generation of defect maps, details the experimental results, and provides further discussions on the nanofabric reconfiguration problem. Finally, we conclude with final remarks and suggestions in Sect. 6.

2 Literature Review

CMOL cell-based, field-programmable gate array (FPGA)-like circuit integrates conventional inverter-based four-transistor MOSFET CMOS cells with uniform reconfigurable nanofabric [10]. Each generic CMOS cell consists of an inverter and two pass transistors. Two-terminal nanodevice "latching switches," which can serve both inter-cells connectivity and wiring logic, are self-assembled at each nanowires cross-point. CMOL cells connectivity is limited to only $M = a^2 - 2$ other cells located within a square-shaped *Connectivity Domain* as shown in dark boarder line in Fig. 1, where a is a positive integer number which indicate the connectivity domain radius.

Conventionally, CMOL cells mapping is split into two steps; 1) defect-unaware cells *Placement* [21], where logic gates are assigned to CMOL cells according to the connectivity domain constraint. 2) *Reconfiguration* around defective components. Further discussion of CMOL circuits mapping design flow is presented in Sect. 3. Recently, several

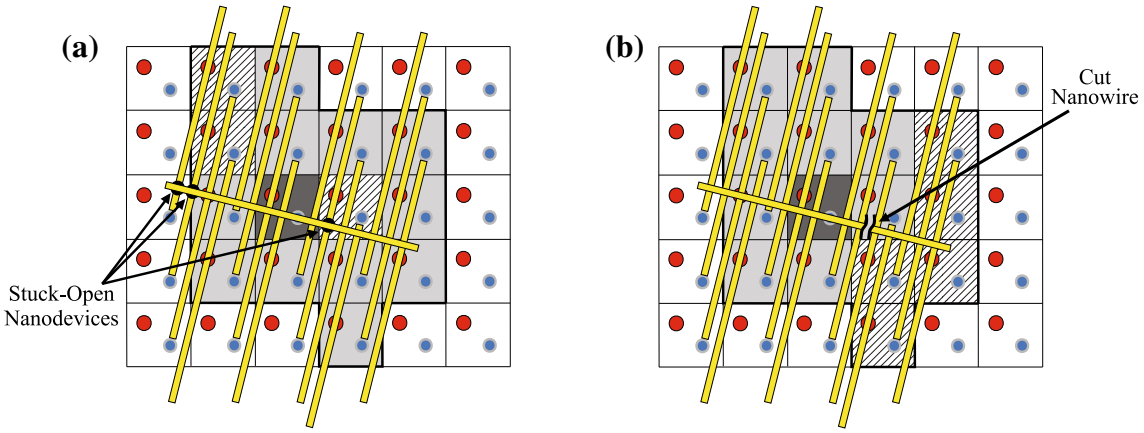

Fig. 1 Defects in CMOL circuits. Cells in *light gray* are the connectivity domain of the cell in *dark gray*. Cells shown in *striped pattern* are not reachable because of defects. **a** Type 1: stuck-open nanodevice defect, **b** Type 2: broken nanowire defect

proposals had been introduced for defect-unaware cell placement/assignment in CMOL circuits including the works of Likharev et al. [15,22] and Hung et al. [23]. Previous attempts to use sub-optimal search heuristics for CMOL placement are reported in [24–27]. Simulated evolution [28] and TS [21,29] heuristics have also been attempted for defect-unaware CMOL placement problem.

Finding the optimal reconfigurations around defects for large nanofabrics is not guaranteed as the given problem is NP-hard [30]. Previously proposed algorithms by Huang et al. [31] and Tahoori et al. [32] are only applicable for small nanofabric crossbars. Greedy algorithms yield degraded results and require considerable computation time in case of high defect rates. Reconfiguration around defective CMOL nanodevices was also reported by Likharev et al. [10]. They developed a reconfiguration algorithm assuming only one type of defect (stuck-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 are realized with the CMOL fabric and are not defective. The authors only showed case studies of reconfiguring Kogge-Stone adder and full crossbar around randomly distributed defects. Other attempt to reconfigure CMOL circuits were reported using SAT solvers [23]. Few clustered defects (around single source) were introduced into small- or medium-size 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 rates of wider range of defects or when circuits' gates have high fan-ins or fan-outs. In such cases, resorting to SAT or sequential-attempts algorithms will be insufficient and circuit reconfiguration may fail. In this paper, we extend our previous work in [33]. We introduce a complete design flow for CMOL cells mapping and introduce a new type of defect. We further elaborate our SimE solution and engineer TS to solve the reconfiguration problem while conducting extensive experiments using different defect maps and scenarios.

3 Design Automation and Problem Description

To implement a combinational circuit in CMOL, NOR-based gates need to be assigned to cells (i.e., CMOL slots). Those cells are connected by two levels of perpendicular nanowires. Programable nanodevices are chemically assembled at the intersection of the two nanowires. In this work, we will consider the assignment of gates to cells in a grid. All cells of the grid and nanodevices may not be defect free. The three used defect models are:

1. Stuck-open: Here the nanodevice connecting two perpendicular nanowires is stuck-open (i.e., not programmable). In this case, the connection between two cells through this nanodevice is not feasible. However, the two CMOL cells can still be used, and only their connectivity domains change.
2. Broken nanowire: An input or output nanowire of a CMOL cell is broken into two segments. Thus, CMOL cell may not be connected to all other CMOL cells within its input/output connectivity domains. The connectivity domains of the affected cells will be significantly reduced.
3. CMOS cell defect: In this case, the CMOS cell could be unusable, because the input/output pins connecting the CMOS stack to the input/output nanowires are broken, or the CMOS inverter is defective. Any cell with this type of defect cannot be used.

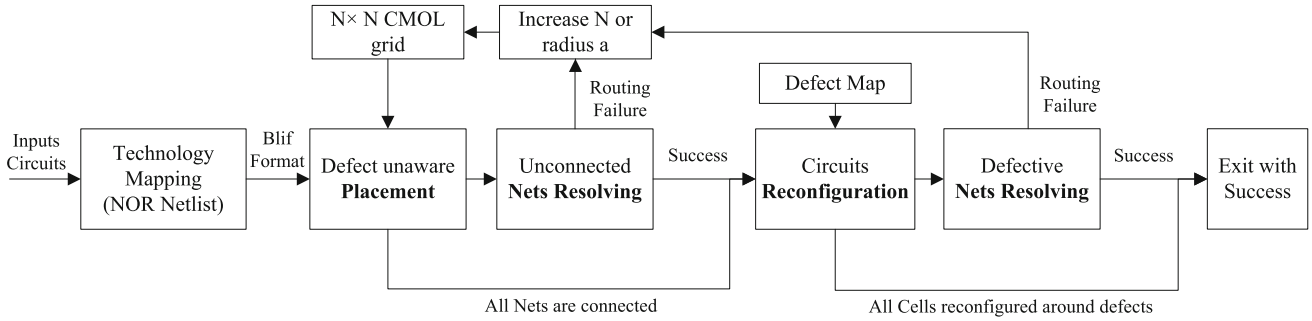

Fig. 2 Design flow of CMOL cells mapping; it consists of two main stages, defect-unaware *Placement* and circuits *Reconfiguration* around defects; each main stage is followed by a complementary *Nets Resolving* step

Defects of type 1 and type 2 are shown in Fig. 1. In broken nanowire defect, each unreachable nanodevice can be modeled as stuck-open. CMOL is susceptible to stuck-close defects; for moderate defect rate, the circuit may become unusable. Likharev et al. [34] reckoned that a defect probability of 0.02 % may not be tolerated. Any probability higher than that means that most of CMOL cells are connected with each others because of the stuck-close nanodevices. For that reason, we did not included stuck-close defects in this work on reconfiguration, as other approaches should be investigated to mitigate this type of defects.

Design automation of CMOL circuits comprises a number of sequential steps. It starts with technology mapping and converting the circuits into a network of NOR/NOT gates (with a certain maximum fan-in). Figure 2 shows the proposed design flow for mapping logic circuits into CMOL grid; it consists of two main stages (*defect-unaware Placement* and *defects Reconfiguration*). When any of the two stages terminate, some connections may still be unresolved (i.e., beyond the connectivity domain in case of placement or use stuck-open or cut nanowires in case of reconfiguration). In such cases, complementary *Nets Resolving* steps are needed to insert extra buffers (i.e., pair of inverters to maintain signal polarity) as intermediate cells between unconnected gates. Those buffers compensate for the missed/defective connections. Each pair of inverters can make a cell connect to another cell, whose distance is within three times the value of the connectivity domain radius a. Violating nets (i.e., those longer than a or use defective nanocomponents) may not be resolved successfully in case no empty cells are available for the additional buffers, or when gates with violating nets are not reachable because all cells in their connectivity domain are occupied. If that is the case, all design stages are repeated using bigger CMOL grid or longer connectivity radius.

We have investigated CMOL placement in our previous works involving SimE [28] and TS [21]. In this work, we address the second design stage, namely reconfiguration and formulate it as an optimization problem. In the subsequent paragraphs, problem constraints are formulated and an objective cost function is given. Iterative heuristics (SimE and TS) are then characterized to seek an optimized cells configuration that minimizes the given cost.

3.1 Problem Formulation

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

$$\forall g_i \in G : L_i \neq 0 \tag{1a}$$

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

where $N(L_i, L_k)$ is a binary value (0 or 1) representing 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. $L = 0$ means the location (i.e., the cell) has a defect of type 3. CMOL reconfiguration is intended to rearrange cells to honor the constraints 1(a) and 1(b), while not violating CMOL connectivity domain constraints defined as follows. For a given gate and its net (g_i, γ_i), any gate g_k in the net γi should satisfy the following.

$$\forall g_i, g_k \in G : (g_i \neq g_k) \Rightarrow (L_i \neq L_k) \quad (2a)$$

$$\forall g_i \in G, \exists g_k \in \gamma_i : L_k \in C(L_i) \quad (2b)$$

where L_k is the location of g_k, $C(L_i)$ is either the input or the output connectivity domain of cell in location L_i. Reconfiguration is highly dependent on defect rates and connectivity radius a. For small connectivity radius, high defect rate may lead to reconfiguration failure.

3.2 Cost Function

The main objective of reconfiguration is to tolerate defects and ensure that the circuit is working properly. The problem we are trying to optimize is to avoid using defective components. Therefore, we should have a measure which can quantify the overall quality of the solution. The overall cost of a solution is expressed as the total number of used defective nanodevices [i.e., the number of connections that violate Eq. 1(b)]. Equation 3 shows the cost of each gate $g_i \in G$ as the number of defective components it uses to connect with its fan-in and fan-out cells. The overall circuit's cost is the sum of individual gates cost.

$$C_i = \sum_{j \in fan\text{-}out_i} u_{i,j} \quad (3a)$$

$$u_{i,j} = \begin{cases} 1 & \text{if } N(L_i, L_j) = 0 \\ 0 & \text{otherwise} \end{cases} \quad (3b)$$

4 Non-deterministic Heuristics Designs

4.1 Solution Representation

Reconfiguration solutions are represented as an arrangement of logic cells in the two-dimensional CMOL layout surface. The layout size corresponds to the number of required CMOL cells to fit each benchmark circuit. Outer cells of the grid are reserved for I/O pins, where I/O pins moves are restricted to these reserved locations. Reconfiguration process starts from a complete cell placement in which all gates are placed within each others connectivity. Each gate is represented by a unique positive integer value. Changing the location of one cell will change the set of nanodevices and nanowires it uses.

4.2 Simulated Evolution

The SimE algorithm is a general search strategy for solving a variety of combinatorial optimization problems. The selection of which elements are 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 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. Further discussion of SimE heuristic can be found in [35].

4.2.1 Goodness Function

In SimE, goodness function is used to evaluate individual elements (i.e., gates) in each generation, where unfit elements are selected and reassigned to other locations. The goodness measure must be strongly related to the objective of the problem, in that sense, the goodness function of each individual element is defined as following:

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

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 fan-ins and fan-outs gates of gate g_i. The above equation assumes a minimization problem (or a maximization of goodness). According to the aforementioned definition, if cell $i's$ connections violate the constraint in Eq. 1(b), the cell will have low goodness value. An example of such cell is shown in Fig. 3, where two defective nanodevices are used to connect gates 4 and 5 with gate i. According to the given definitions

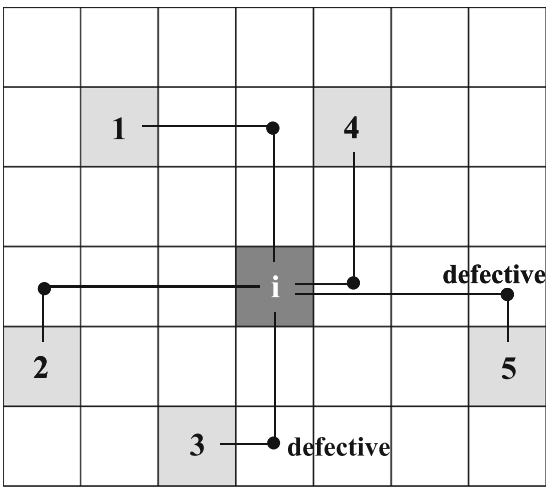

Fig. 3 Evaluation of gate i's goodness; connection between cell i and cells 4 and 5 use defective nanodevices, $goodness_i = connect_i/|\gamma_i| = 3/5 = 0.6$. Nanodevices are shown as *black dots*

of the goodness function, the value of $|\gamma_i|$ do not change from generation to generation. It is only computed once and is based on the original circuit description.

4.2.2 Selection Function

It uses original SimE selection function [35], an element (i.e., 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 its chance of retaining its current location. While lower the goodness value, the more likely the element will be perturbed and reallocated. SimE selection function has a nondeterministic nature; an individual with a high goodness (i.e., close to one) still has a nonzero probability of being selected. This stochastic role gives SimE the hill climbing property. Reallocating the selected elements can be done in a deterministic order that is correlated with the objective function being optimized. Hence, prior to the allocation step, the elements in the selection set are sorted in an ascending order based on their netlist size, where elements with higher cardinality of γ_i are processed first. Selection function for CMOL reconfiguration is similar to the one used for defect-unaware placement [28]. The only difference is in the way cells goodness is evaluated. In defect-unaware placement, goodness is based on the number of unconnected cells, while the goodness function in reconfiguration phase is based on the number of used defective components.

4.2.3 Allocation Function

This function has the most impact on the quality of the solution; it is intended to generate a new solution that is inherently better than the old one. The design of the allocation function is related to the problem specifications. The allocation function is a complex form of genetic mutation, it alters the locations of all elements in the selection set one after the other. The function is fully aware of the presence of defects. It actually, swap cells based on the cost defined by the number of used defective nanodevices. For each selected element, the allocation function evaluates the cost of swapping the element with another one in the grid based on the cost function in Eq. 3a. Then, the best swap is chosen. This constitute a global allocation policy, which could prove to be very useful specially in the early iterations. Unlike the allocation function in the defect-unaware placement [28], an additional constraint applies for gates movements during reconfiguration; the reallocation of gates is constrained to the region defined by the intersection of the connectivity domains of the two cells where the gates are located and their fan-in and fan-out cells. This ensures that reconfiguration do not invalidate the assignment made by the placement phase (i.e., does not move cells in which some of the connections violate the constraints in Eq. 2(b)). Furthermore, CMOL cells with stuck-open or cut nanowires defects can still be used in the reconfiguration stage. SimE's allocation function can assign gates to those cells as long as the assigned gate's fan-ins and fan-outs do not require the use of defective complements.

An evaluation of SimE implementation is shown in Fig. 4a, where the number of elements selected for perturbation in each iteration is given. The size of the selection set decreases with time, until a final solution with the least cost is reached. This is also an indication that the overall quality of solution is improving. In addition, Fig. 4b shows the problem cost (Eq. 3) of the solution in each iteration. It shows how the heuristic is evolving to better solutions without being too greedy.

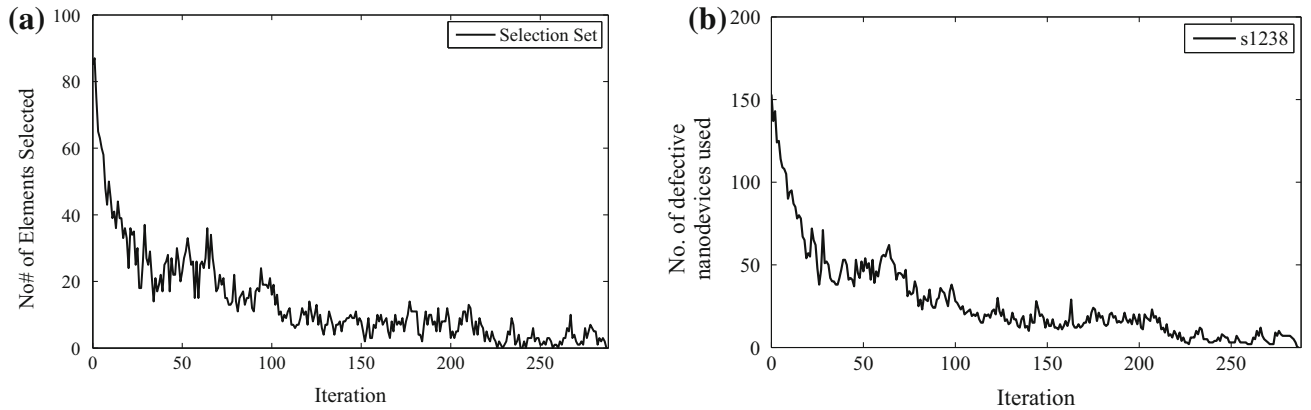

Fig. 4 SimE reconfiguration heuristic. **a** Illustration of change in the size of the selection set with iterations for the SimE reconfiguration heuristic for s1238.blif benchmark. Selection set size decreases with time, until a final solution is found, **b** change in number of defective cells due to reconfiguration. Cost per iteration in SimE run shown for s1238.blif benchmark. Solutions found by SimE are evolving to better ones without being greedy

4.3 Tabu Search

Tabu Search algorithm is a non-deterministic iterative heuristic that has been applied to solve many combinatorial optimization problems. It is considered as a generalization of local search algorithms. At each step, TS explore the local neighborhood of the current solution and the best solution in that neighborhood is selected as the new current solution. TS is a heuristic that proceeds by making iterative perturbations while preventing cycling to certain number of recently visited points in search space. Detailed description of TS's attributes and constructs can be found in [35]. In engineering TS for reconfiguration phase, we have used the same methodology, move attributes and aspiration criteria used in defect-unaware placement [21]. Reconfiguration phase neighborhood is generated by performing one move (i.e., perturbation). Each solution in the candidate list is evaluated based on the change in number of used defective components before and after the swap, taking care that swaps do not violate the constraints set by Eq. 2(b). The actual size of the candidate list is empirically set based on the performance of the heuristic and the problem behavior. Figure 5(a) shows the quality of the solutions produced by TS over iterations. The heuristic steadily converges to near optimal solution.

Figure 5(b) shows the change in the problem cost with respect to changes in candidate list sizes for different defect rates. Each rate constitutes a distinct instance of the reconfiguration problem. In high defect rates, a small candidate list (e.g., between 5 and 35) results in bad solutions, whereas for low rates, a small list is sufficient. The problem becomes more constrained when many nanodevices are defective; thus, TS requires more choices to effectively explore the search space. Throughout our implementation, we have used different list sizes for different defect rates; a maximum value of 60 is used as an upper bound limit on the list size. We have further used the same candidate list probabilistic swap selection described in [21] using Gaussian random variable that has mean $m0 = 0$ and standard deviation $\sigma = circuit-size$. This modified neighborhood generation has reduced the candidate list sizes considerably compared with that required when random cells are selected.

4.4 Nets Resolving Procedure (Buffers Insertion)

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. Intermediary inverters are inserted between cells which have their connections faulty. Buffers insertion is done by performing the same SimE or TS heuristics but with some modifications. The heuristics will assign pairs of inverters to empty locations (i.e., unused CMOL cells) so that two cells with defective interconnect become connected. Those buffers compensate for the missed/defective connections. Each pair of inverters can make a cell connect to another cell using a new set of nanodevices and nanowires. The constraints highlighted in Eqs. 1 and 2 should also apply for the inserted inverters. SimE allocation function and TS swaps are modified as to only permit the interchange of two inserted inverters or to reallocate an inverter to a different unused CMOL cell. The other already occupied CMOL cells are assumed to be fixed and no perturbation can involve any one of them.

Initially, the *Net Resolving* procedure starts with one buffer (pair of inverters) for each defective inter-cells connection. Inverters are randomly assigned and then SimE or TS are called with the aforementioned modifications. The heuristics then continuously try to improve the locations of the inserted inverters until all of them are placed and all defective nets

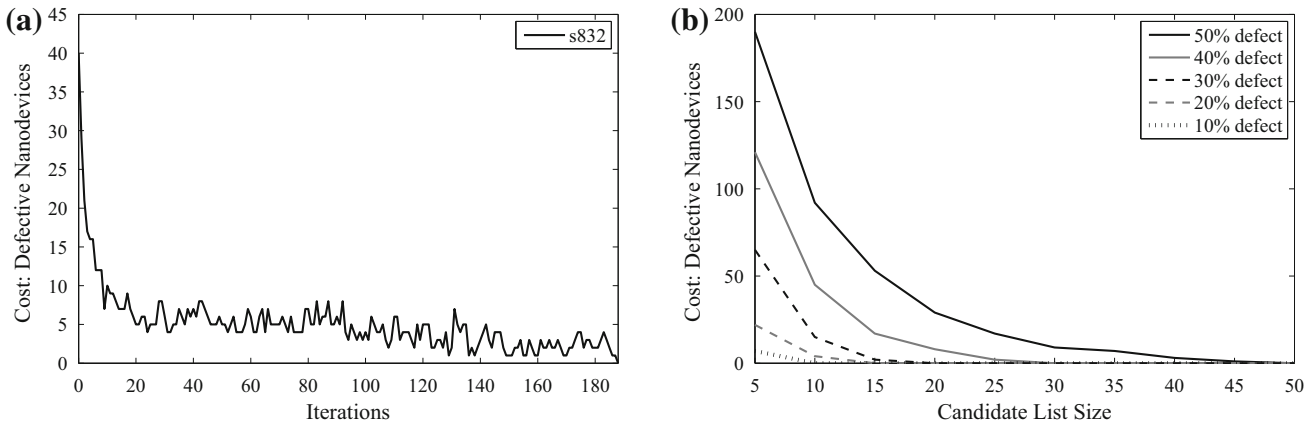

Fig. 5 Tabu search reconfiguration heuristic. **a** Change of reconfiguration cost per iteration for s832.blif benchmark, **b** change of reconfiguration cost with respect to changes in candidate list size for different nanodevices stuck-open defect rates $q_{\text{nano}} = 10$–$50\,\%$. High defect rates require larger candidate list sizes. s1196.blif benchmark

are resolved. If Net Resolving procedure terminates while still some connections are defective (i.e., the heuristics failed to substitute them by inserting additional buffers), the unresolved interconnects are substituted by two buffers instead of one (i.e., four inverters) and the procedure is repeated. This process goes on until all violating connections are resolved by new buffers that respect CMOL constraints, or until no cells are available in the grid to which inverters can be assigned to. In our implementation, we limited the number of inverters that can be inserted to substitute a particular defective interconnect to 6 to avoid major deterioration of the circuits timing delay.

5 Experimental Results

In this section, we describe the experimental setup and final results of CMOL hybrid nanofabric reconfiguration. Next, defect scenarios, benchmark circuits and simulation environment are outlined. Results yielded by SimE and TS are presented afterward. Final solutions (i.e., reconfigured CMOL circuits) are tested and verified to insure the correctness of the whole design stages. Active nanodevices are cross-checked with defect information stored in the defect maps, and randomly generated input patterns are applied to test circuits' functionality.

5.1 Defect Maps

Different methods for simulating faults distribution have been reported in the literature [36]. In this work, two methods are used for stuck-open faults simulation. In the first approach, a uniform random distribution is used. For any given nanodevice, a random number p with uniform distribution between 0 and 1 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:

$$pmf(x, y) = Ce^{-\frac{(x-x_0)^2+(y-y_0)^2}{2\sigma^2}} \tag{5}$$

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 defect clusters. The value of constant C is related to grid size and nanodevices density. For each nanodevice, we generate a random number p between 0 and 1. A fault is injected if $p \leq pmf(x, y)$. While other approaches and mathematical distributions exist for simulating clustered faults, we have chosen this approach as it is more prominently applied in literature [37].

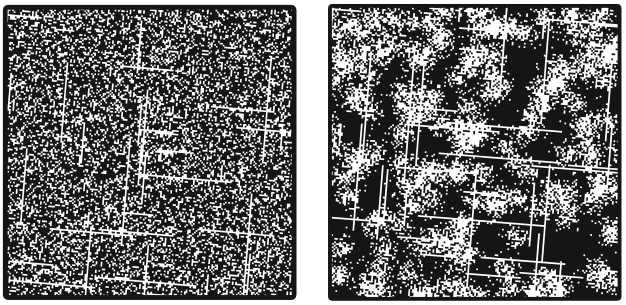

Fig. 6 Defect maps: **a** randomly distributed defects map, **b** clustered defects map when $C = 0.7$ and $\sigma = \frac{a}{3}$. Nanodevices stuck-open defect rate $q_{\text{nano}} = 30\,\%$ and nanowires cut rate $q_{\text{wire}} = 10\,\%$. *White dots* represent non-programmable nanodevices (i.e., stuck-open), while *black dots* represent programable ones

For broken nanowires defects, 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-open. In similar manner, CMOS cells are assumed to be defective based on a defect rate q_{cell}. Figure 6 shows defect maps for stuck-open defect rate $q_{\text{nano}} = 30\,\%$ and wires cut rate $q_{\text{wire}} = 10\,\%$. The first map shows randomly distributed defects and the second shows clustered defects when $C = 0.7$ and $\sigma = \frac{a}{3}$. White dots represent non-programmable nanodevices (i.e., stuck-open), while black dots represent programable ones.

5.2 Experimental Setup

Given defects distributions discussed in Sect. 5.1, we have performed multiple experiments using three types of defect maps: a randomly generated map ($R1$) and two clustered maps ($C1$ and $C2$). Those maps have $C = 0.8$ and standard deviation $\sigma = \frac{2a}{3}$ and $\sigma = \frac{4a}{3}$ for $C1$ and $C2$, respectively. C and σ constants were chosen through experimentation by trying different values and observing the effect on defect density. The values were chosen to correspond with the grid sizes of the circuits under test and the connectivity domain radius a. Evaluation of search heuristics efficiency and performance is done using the three defect scenarios shown in Table 1, where q_{nano} is the probability of a nanodevice is stuck-open (type 1 defect), q_{wire} is the probability of a nanowire is broken (type 2 defect), and q_{cell} is the probability of a CMOS cell is defective (type 3 defect). Scenario (i) includes five experiments when q_{nano} ranges between 10 and 50 %, while $q_{\text{wire}} = 20\,\%$ and $q_{\text{cell}} = 0\,\%$, and similarly scenario (ii) is comprised of seven experiments when q_{wire} probability is between 10 and 70 % and $q_{\text{nano}} = 20\,\%$. Scenario (iii) has

Table 1 Defect scenarios; scenario (i) includes 5 experiments for different q_{nano} probabilities, scenario (ii) includes seven experiments for different q_{wire} probabilities, and scenario (iii) includes two experiments for different q_{cell} probabilities

Scenario	q_{nano} (%)	q_{wire} (%)	q_{cell} (%)
(i)	10–50	20	0
(ii)	20	10–70	0
(iii)	20	20	10–20

fixed q_{nano} and q_{wire} probabilities, and q_{cell} equals to 10 or 20 %.

Reconfiguration is applied to 19 circuits of different sizes from ISCAS'89 benchmarks suite [38]. Further consideration should be given to those benchmarks by replacing sequential elements (i.e., flip-flop) inputs and outputs with POs and PIs, respectively. Circuits are then mapped by SIS synthesis tool [39] to a NOR netlist with maximum of five inputs. Details of ISCAS'89 circuits are shown in Table 2; the numbers of $Cells$ including $Gates$, $Inputs$ and $Outputs$ are given.

Table 2 also gives CMOL 2D grids sizes, which are used to implement ISCAS'89 benchmarks; *Area* ($Row \times Column$) indicates the number of *Rows* and *Columns* and the number of *CMOL Cells* which a given benchmark uses. The grid sizes given under $q_{\text{cell}} = 0$ % are those for circuits reconfiguration in defect scenarios (i) and (ii). For scenario (iii), the used area is shown under $q_{\text{cell}} = 10$ % and $q_{\text{cell}} = 20$ %. Defect maps $R1$, $C1$ and $C2$ are generated for each grid size; benchmarks that need similar CMOL gird sizes are reconfigured using same defect maps. For each benchmark circuit, the grid utilization is shown in the table; $GU\%$ represents the ratio of used cells in CMOL gird.

Simulated evolution and TS are implemented using Java programming language and run on a machine with 1.5 GHz Intel Pentium M processor and 512 MB memory. Technology mapping is achieved using SIS logic synthesis tool [39]. Circuits verification and defect maps generation programs are also written in Java. Comparisons between CMOL reconfigured circuits and original benchmarks logical functionality are carried out by HOPE simulator [40], which runs on a LINUX machine. Heuristics are regulated to stop when solution's cost (i.e., number of defective nanodevices) becomes zero or when a predefined number of iterations is reached; in our case, we have set the number of iterations in SimE to 4000 and used a larger number of iterations for TS. The average value of results obtained from 20 successful reconfigurations for each circuit is reported, where each run uses different seeds for random numbers.

Table 2 ISCAS'89 benchmarks and corresponding CMOL's grid sizes in ($row \times column$), and grid utilization GU%, given different CMOS cells defect (q_{cell}) probabilities

Circ.	Cells	Gates	Inputs	Outputs	$q_{\text{cell}} = 0$ %		$q_{\text{cell}} = 10$ %		$q_{\text{cell}} = 20$ %	
					Area ($row \times column$)	GU%	Area ($row \times column$)	GU%	Area ($row \times column$)	GU%
s27	19	8	7	4	25 (5 × 5)	76.00	–	–	–	–
s208	136	109	18	9	169 (13 × 13)	80.47	–	–	–	–
s298	122	85	17	20	144 (12 × 12)	84.72	–	–	–	–
s344	180	130	24	26	196 (14 × 14)	91.84	–	–	–	–
s349	184	134	24	26	196 (14 × 14)	93.88	–	–	–	–
s382	175	124	24	27	196 (14 × 14)	89.29	–	–	–	–
s386	164	138	13	13	196 (14 × 14)	83.67	–	–	–	–
s400	188	137	24	27	196 (14 × 14)	95.92	–	–	–	–
s420	299	248	34	17	361 (19 × 19)	82.83	–	–	–	–
s444	187	136	24	27	196 (14 × 14)	95.41	–	–	–	–
s510	304	266	25	13	361 (19 × 19)	84.21	–	–	–	–
s526	273	222	24	27	324 (18 × 18)	84.26	–	–	–	–
s641	302	206	54	42	676 (26 × 26)	44.67	676 (26 × 26)	44.67	676 (26 × 26)	44.67
s713	321	225	54	42	676 (26 × 26)	47.49	676 (26 × 26)	47.49	676 (26 × 26)	47.49
s820	447	400	23	24	529 (23 × 23)	84.50	576 (24 × 24)	77.60	625 (25 × 25)	71.52
s832	454	407	23	24	529 (23 × 23)	85.82	576 (24 × 24)	78.82	625 (25 × 25)	72.64
s838	606	507	66	33	676 (26 × 26)	89.64	729 (27 × 27)	83.13	784 (28 × 28)	77.30
s1196	675	613	31	31	729 (27 × 27)	92.59	841 (29 × 29)	80.26	900 (30 × 30)	75.00
s1238	724	662	31	31	784 (28 × 28)	92.35	900 (30 × 30)	80.44	961 (31 × 31)	75.34

Table 3 Reconfiguration CPU computation time for defect scenario (i) and defect maps $R1$, $C1$ and $C2$

Maps	q_{nano} (%)	SimE		TS	
		Max	Avg.	Max	Avg.
R1	10	0.51	0.11	1.70	0.39
	20	0.96	0.20	3.81	0.83
	30	2.46	0.41	17.76	2.99
C1	10	0.64	0.14	1.28	0.39
	20	1.38	0.25	3.20	0.79
	30	2.72	0.52	44.96	5.27
C2	10	0.51	0.15	2.08	0.56
	20	1.15	0.25	4.54	1.08
	30	2.82	0.50	44.19	5.97

Max., maximum CPU computation time; Avg., average CPU computation time; q_{nano}, probability of stuck-open nanodevices' defects ($q_{\text{nano}} = 10$–$30\,\%$—$q_{\text{wire}} = 20\,\%$—$q_{\text{cell}} = 0\,\%$)

5.3 Reconfiguration Results

This section presents the final results of CMOL reconfiguration using SimE and TS iterative heuristics. To reconfiguring circuits in CMOL, we have adhered to the original description of the connectivity domain shown in Fig. 1 with connectivity radius $a = 18$.[1] In SimE, we have used a bias B between $[-0.06, 0.05]$, where the small bias values are used for high defect rates. Negative bias was required to reduce the number of selected elements (especially in early iterations) to prevent the heuristic from performing conflicting moves, which may result in poor exploration of the search space. Results reported in this section were not compared with those based on Satisfiability [23] (discussed in Sect. 2), 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 3 report the maximum and averaged CPU computation times (in seconds), which SimE and TS needed to reconfigure all benchmark circuits for random and clustered defect maps when q_{nano} is between 10 and 30 %. For this given range of stuck-open probabilities, both SimE and TS were successful in reconfiguring circuits around all defective components. The table shows that SimE required less computation time than TS as it employs an evolutionary goodness and selection functions. For example, in clustered defect map (C2) and for $q_{\text{nano}} = 30\,\%$, SimE average computation time is equal to 0.50 whereas TS average time is 5.97. 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 4 shows SimE results for nanodevice stuck-open defect probabilities $q_{\text{nano}} = 40$–$50\,\%$. *Time* is CPU computation time in seconds and *Buffers* is the number of inserted buffers to resolve defective nets. For $q_{\text{nano}} = 40\,\%$, all circuites were reconfigured successfully, while for $q_{\text{nano}} = 50\,\%$, only s820 and s832 needed additional buffers to resolve defective nets which SimE was unable to reconfigure. Those two circuits also needed more buffers when defects are clustered compared with randomly distributed defects (e.g., s832 required three buffers in $R1$, and six in $C1$ and $C2$). Moreover, SimE's reconfiguration of clustered defects consumed more CPU time compared with reconfiguration of random defects. For example, $C1$ and $C2$ average CPU time is 3.11, 3.37 s, respectively, compared with 2.01 s for random map $R1$.

Similarly, Table 5 gives TS results; the heuristic found more costly results than those reported by SimE for both 40 and 50 % probabilities. Furthermore, TS found its solutions in considerably more computation time than SimE. TS has failed to reconfigure s820 and s832 around clustered defects (i.e., maps $C1$ and $C2$) when $q_{\text{nano}} = 50\,\%$. The high-density clustered defects rendered search space exploration difficult leading to many unresolved nets, which make adding additional buffers using *Net Resolving* procedure not feasible.

Further, we have investigated the robustness of our heuristics designs by reconfiguring benchmark circuit s1238 using 20 different clustered defect maps (i.e., 20 clustered defect maps all for grid size of 28×28). All defect maps have $C = 0.8$, $\sigma = \frac{4a}{3}$, defect rate $q_{\text{nano}} = 50\,\%$ and cut rate $q_{\text{wire}} = 20\,\%$. The heuristics were run for 40 times for each map; reconfiguration was successful in 19 out of 20 maps, with overall successful reconfiguration rate of 60 % (i.e., 60 % of the 40 run $\times$ 20 maps = 800 run). For defect maps with defect rate $q_{\text{nano}} < 50\,\%$, the heuristics successfully reconfigured all of the 20 defect maps, and the overall successful reconfiguration rate was over 95 %.

Reconfiguration results for defect scenario (ii) are shown in Tables 6 and 7. The reconfiguration of two circuits (i.e., $s820$ and $s1238$) is performed when up to 70 % of the nanowires are cut and 20 % of the nanodevices are stuck-open. For both circuits, SimE has found successful reconfigurations without the need for any additional buffers even when the probability of broken wires is as high as 70 %. While on the contrary, results found by TS required

[1] Previous solutions for CMOL cells placement [21,23–26,28] defined the connectivity domain based on Manhattan distance, where a cell is said to be within the connectivity domain of another cell if the Manhattan distance between them is less or equal to connectivity radius $a = 12$. Here in this paper, we have defined the connectivity domain based on the original description given in [22] as shown in Fig. 1. Given a particular connectivity radius a, the former definition has bigger connectivity domains than the later one. For that reason, we have used a connectivity radius $a = 18$ to compensate the difference between the two definitions. (e.g., if $a = 4$ the size of the connectivity domain shown in Fig. 1 will be $M = a^2 - 2 = 14$ cells, while if Manhattan distance is used $M = 40$).

Table 4 Simulated evolution reconfiguration results for defect scenario (i) and defect maps $R1$, $C1$ and $C2$, when $q_{\text{nano}} = 40$–$50\,\%$—$q_{\text{wire}} = 20\,\%$–$q_{\text{cell}} = 0\,\%$

Circ.	$R1$				$C1$				$C2$			
	$q_{\text{nano}} = 40\,\%$		$q_{\text{nano}} = 50\,\%$		$q_{\text{nano}} = 40\,\%$		$q_{\text{nano}} = 50\,\%$		$q_{\text{nano}} = 40\,\%$		$q_{\text{nano}} = 50\,\%$	
	Time	Buffers	Time	Buffers	Time	Buffers	Time	Buffers	Time	Buffers	Time	Buffers
s27	0.03	0	0.03	0	0.03	0	0.03	0	0.03	0	0.03	0
s208	0.03	0	0.03	0	0.03	0	0.03	0	0.06	0	0.03	0
s298	0.06	0	0.06	0	0.06	0	0.19	0	0.03	0	0.29	0
s344	0.03	0	0.06	0	0.06	0	0.10	0	0.06	0	0.10	0
s349	0.06	0	0.10	0	0.10	0	0.10	0	0.10	0	0.13	0
s382	0.10	0	1.22	0	0.26	0	0.32	0	0.19	0	1.28	0
s386	0.26	0	3.78	0	0.80	0	3.10	0	0.48	0	8.54	0
s400	0.13	0	0.64	0	0.32	0	0.64	0	0.29	0	0.48	0
s420	0.16	0	0.32	0	0.16	0	0.26	0	0.19	0	0.22	0
s444	0.16	0	0.77	0	0.32	0	0.35	0	0.26	0	0.74	0
s510	1.02	0	1.09	0	0.90	0	2.34	0	1.63	0	2.21	0
s526	1.06	0	2.21	0	0.90	0	1.06	0	2.18	0	6.34	0
s641	0.29	0	0.61	0	0.32	0	1.22	0	0.32	0	1.06	0
s713	0.38	0	0.64	0	0.38	0	1.79	0	0.32	0	0.61	0
s820	4.26	0	8.06	3	5.44	0	12.13	4	4.19	0	11.78	4
s832	6.50	0	9.18	3	6.50	0	18.27	6	7.07	0	15.84	6
s838	0.90	0	1.31	0	1.02	0	1.79	0	1.22	0	1.82	0
s1196	0.99	0	3.04	0	2.62	0	6.24	0	2.75	0	7.01	0
s1238	0.99	0	4.99	0	3.68	0	9.12	0	7.62	0	5.54	0
Avg.	0.92	0	2.01	0	1.26	0	3.11	1	1.53	0	3.37	1

Time: CPU processing time
Buffers: the number of inserted buffers to resolve defective nets
q_{nano}: probability of Stuck-Open nanodevices' defects

the insertion of buffers in order to have a functional and defect-free circuits. For high wires cut rates, TS failed to reconfigure s820 circuit, while it was successful for s1238. Solution reported when defective nanodevices are randomly distributed are better than those when nanodevices defects are clustered. In both SimE and TS, computation time increases when the problem become more complex (i.e., q_{wire} increases and nanodevices defects are clustered). When many nanowires are cut, CMOL cells connectivity domains become considerably limited. Furthermore, clustered defects means that some particular CMOL cells have many cells in their connectivity domains unreachable.

Defect scenario (iii) results are shown in Table 8. Defective CMOS cells bring about a constraint for both *Placement* and *Reconfiguration* design stages. To investigate this defect type, we have run SimE to place logic gates into defect-free CMOS cells and then reconfiguration rearranged gates location to overcome nanofabric defects without relying on CMOS defective cells. In this scenario, *Placement* follows the formulation given in our previous work [28] along with the constraint in Eq. 1a. SimE has found solutions with zero buffers for all circuits under test except for circuit $s820$, which required two buffers for $q_{\text{cell}} = 10\,\%$ and three buffers for $q_{\text{cell}} = 20\,\%$. The given $Time$ in Table 8 is the CPU time for all design stages (i.e., for *Placement*, *Reconfiguration* and *Net Resolving*).

In comparing SimE and TS heuristics performance and results, it is clear that SimE is finding better results in less computation time for both random and clustered defects. That is attributed to the fact that SimE is applying an evolutionary search space exploration in which cells that are ought to be moved are selected based on goodness evaluation technique. Although TS is a widely adopted search heuristic, but still it has a shortcoming when circuits rely on multiple inter-cells interconnect and in case of high nanofabric defect rates.

5.4 Solutions Verification

Each logic circuit implemented in CMOL uses a number of nanodevices to connect its modules (i.e., gates). The final outcome of CMOL cell mapping heuristics is the list of defect-free nanodevices that should be programmed (i.e., set

Table 5 Tabu Search reconfiguration results for defect scenario (i) and defect maps $R1$, $C1$ and $C2$, when $q_{\text{nano}} = 40$–$50\,\%$—$q_{\text{wire}} = 20\,\%$—$q_{\text{cell}} = 0\,\%$

Circ.	$R1$				$C1$				$C2$			
	$q_{\text{nano}} = 40\,\%$		$q_{\text{nano}} = 50\,\%$		$q_{\text{nano}} = 40\,\%$		$q_{\text{nano}} = 50\,\%$		$q_{\text{nano}} = 40\,\%$		$q_{\text{nano}} = 50\,\%$	
	Time	Buffers	Time	Buffers	Time	Buffers	Time	Buffers	Time	Buffers	Time	Buffers
s27	0.03	0	0.03	0	0.03	0	0.03	0	0.03	0	0.03	0
s208	0.10	0	0.22	0	1.60	0	9.82	0	0.96	0	1.09	0
s298	0.22	0	1.28	0	0.26	0	4.32	0	0.19	0	5.25	0
s344	0.19	0	1.92	0	0.93	0	3.94	0	0.42	0	4.51	0
s349	0.38	0	6.43	0	3.30	0	29.95	0	0.93	0	26.75	0
s382	0.61	0	5.60	0	1.60	0	5.41	0	1.28	0	11.94	0
s386	2.43	0	69.95	1	21.06	0	69.38	1	22.21	0	65.76	2
s400	1.50	0	15.36	0	5.09	0	15.84	0	4.80	0	25.63	0
s420	0.64	0	1.50	0	0.64	0	2.46	0	0.90	0	2.46	0
s444	3.68	0	9.50	0	12.99	0	7.30	0	7.52	0	16.54	0
s510	12.64	0	85.98	1	11.90	0	80.48	1	18.62	0	70.40	1
s526	4.19	0	72.67	1	4.70	0	61.22	1	8.64	0	72.70	1
s641	2.21	0	37.38	1	2.66	0	37.15	1	4.13	0	46.18	2
s713	2.05	0	41.47	1	2.30	0	45.22	2	2.88	0	5.12	3
s820	97.98	2	160.16	8	90.05	1	–	–	88.19	1	–	–
s832	103.14	2	171.87	7	98.53	2	–	–	92.42	1	–	–
s838	3.04	0	11.10	0	4.38	0	23.14	0	5.50	0	63.84	0
s1196	57.38	0	99.26	0	48.26	0	136.93	0	52.29	0	133.25	0
s1238	33.22	0	153.54	0	49.50	0	184.06	3	98.24	0	156.03	0
Avg.	17.14	0	49.75	1	18.94	0	42.16	1	21.59	0	41.62	1

Time: CPU processing time
Buffers: the number of inserted buffers to resolve defective nets
q_{nano}: probability of Stuck-Open nanodevices' defects

Table 6 SimE and TS results for s820 circuit's reconfiguration around cut nanowires (scenario ii), when $q_{\text{nano}} = 20\,\%$–$q_{\text{cell}} = 0\,\%$

q_{wire} (%)	Simulated evolution						Tabu Search					
	$R1$		$C1$-$\sigma = \frac{2a}{3}$		$C2$-$\sigma = \frac{4a}{3}$		$R1$		$C1$ -$\sigma = \frac{2a}{3}$		$C2$ -$\sigma = \frac{4a}{3}$	
	Time	Buffers	Time	Buffers	Time	Buffers	Time	Buffers	Time	Buffers	Time	Buffers
10	0.29	0	0.29	0	0.45	0	1.38	0	1.25	0	3.65	0
20	0.42	0	0.48	0	0.77	0	4.29	0	2.88	0	5.79	0
30	1.15	0	1.12	0	1.98	0	12.80	0	9.22	0	71.81	1
40	1.34	0	2.59	0	4.13	0	25.28	0	92.48	1	99.81	2
50	2.56	0	10.11	0	9.79	0	119.97	1	134.50	2	111.55	3
60	7.20	0	13.47	0	18.30	0	150.72	2	140.67	3	–	–
70	13.60	0	26.14	0	22.85	0	190.21	2	–	–	–	–
Avg.	3.79	0	7.74	0	8.32	0	72.09	1	63.50	1	58.52	1

Time: CPU processing time
Buffers: the number of inserted buffers to resolve defective nets
q_{wire}: probability of nanowires cut

to "ON" state) and the list of defect-free CMOS cells. Completely mapped and reconfigured CMOL circuits at the end of our proposed design flow are tested and verified following the steps shown in Fig. 7.

Verification procedure starts by checking whether any of the active nanodevices used to connect CMOL cells is defective. This is done by cross-matching the list of used nanodevices with defect information stored in the defect

Table 7 Particle SimE and TS results for s1238 circuit's reconfiguration around cut nanowires (scenario ii), when $q_{\text{nano}} = 20\,\%$—$q_{\text{cell}} = 0\,\%$

q_{wire} (%)	Simulated evolution						Tabu Search					
	$R1$		$C1$ -$\sigma = \frac{2a}{3}$		$C2$-$\sigma = \frac{4a}{3}$		$R1$		$C1$-$\sigma = \frac{2a}{3}$		$C2$-$\sigma = \frac{4a}{3}$	
	Time	Buffers	Time	Buffers	Time	Buffers	Time	Buffers	Time	Buffers	Time	Buffers
10	0.38	0	0.45	0	0.61	0	1.5	0	2.34	0	3.52	0
20	0.58	0	0.54	0	1.22	0	2.27	0	3.58	0	6.05	0
30	0.83	0	0.86	0	0.9	0	3.78	0	5.28	0	4.93	0
40	0.86	0	1.73	0	1.76	0	5.92	0	9.6	0	8.38	0
50	2.21	0	1.95	0	1.7	0	9.38	0	14.5	0	8.74	0
60	3.46	0	3.04	0	4.99	0	16.13	0	26.21	0	32.96	0
70	4.77	0	5.54	0	6.14	0	35.04	0	47.36	0	38.18	0
Avg.	1.87	0	2.02	0	2.47	0	10.57	0	15.55	0	14.68	0

Time: CPU processing time
Buffers: the number of inserted buffers to resolve defective nets
q_{wire}: probability of nanowires cut

Table 8 Implementation of SimE for defect scenario (iii), when $q_{\text{nano}} = 20\,\%$—$q_{\text{wire}} = 20\,\%$

Circ.	$q_{\text{cell}} = 10\,\%$		$q_{\text{cell}} = 20\,\%$	
	Time	Buffers	Time	Buffers
s641	17.06	0	18.50	0
s713	23.17	0	28.22	0
s820	136.96	2	240.13	3
s838	60.48	0	95.04	0
s1196	295.68	0	320.90	0
s1238	390.88	0	415.30	0
Avg.	154.04	0	186.35	1

Time: CPU processing time.
Buffers: the number of inserted buffers to resolve defective nets.
q_{cell}: probability of CMOS cells defects

map of the given CMOL gird. Then, the circuit netlists are reconstructed per the description of the nanodevices list and the circuit is written in. bench formate. The original circuit description (i.e., the one before mapping) and the reconstructed. bench file are forwarded to HOPE simulator [40] along with randomly generated input patterns. We use perl script to run the simulator and compare the output of the two circuits to decide whether they match. Based on the generated outputs, we conclude whether the two circuits are functionally equivalent or not. The verification procedure make sure that our heuristics perturbations do not change the circuit description and verify that mapped circuits have the same logical functionality as the original ones.

6 Conclusion

In this paper, we presented a design automation flow for reconfiguration-based defect-tolerant in nanofabric architectures. SimE and TS are utilized to optimize and reconfigure defective CMOL circuits. We analyzed the problem behavior and engineered heuristic solutions that exploit better understanding of the limitations imposed by CMOL connectivity domain and defective components. Further, we analyzed the heuristics performance and tuned their parameters. Results obtained showed that circuits can be reconfigured to become functional even if 50 % of CMOL nanodevices are stuck-open or if up to 70 % of the architecture's nanowires are cut. Our findings show that reconfiguration is an effective

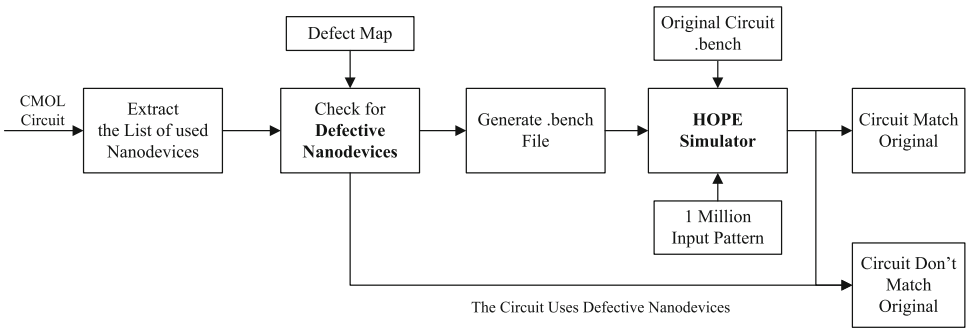

Fig. 7 Verification steps: Mapped circuits are cross-matched with defect map information. The mapped and original circuits are simulated using HOPE simulator and outputs compared to decide whether circuits match

defect-tolerance technique for the emerging nanofabric-based systems, and nanofabric's imprecise assembly can be negated by elaborate CAD tools.

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

References

1. Frank, D.J.; Dennard, R.H.; Nowak, E.; Solomon, P.M.; Taur, Y.; Philip Wong, H.-S.: Device scaling limits of Si MOSFETs and their application dependencies. Proc. IEEE **89**(3), 259–288 (2001)
2. Konstantin, L.: Electronics below 10 nm. In: Greer, J.; Korkin, A.; Labanowski J. (eds.) Nano and Giga Challenges in Microelectronics, pp. 27–68. Elsevier, Amsterdam. http://dx.doi.org/10.1016/B978-044451494-3/50002-0 (2003)
3. Park, H.; Park, J.; Lim, A.K.L; Anderson, E.H.; Alivisatos, A.P.; McEuen, P.L.: Nanomechanical oscillations in a single-C60 transistor. Nature **407**(6800), 57–60 (2000)
4. Kubatkin, S.; Danilov, A.; Hjort, M.; Cornil, J.; Bredas, J.-L.; Stuhr-Hansen, N.; Hedegard, P.; Bjornholm, T.: Single-electron transistor of a single organic molecule with access to several redox states. Nature **425**(6959), 698–701 (2003)
5. Tougaw, P.D.; Lent, C.S.: Logical devices implemented using quantum cellular automata. J. Appl. Phys. **75**(3), 1818–1825 (1994)
6. Collier, C.P.; Wong, E.W.; Belohradsky, M.; Raymo, F.M.; Stoddart, J.F.; Kuekes, P.J.; Williams, R.S.; Heath, J.R.: Electronically configurable molecular-based logic gates. Science **285**(5426), 391–394 (1999)
7. Chen, J.; Reed, M.A.; Rawlett, A.M.; Tour, J.M.: Observation of a large on-off ratio and negative differential resistance in an electronic molecular switch. Science **286**, 1550–1552 (1999)
8. Zhirnov, V.V.; Herr, D.J.C.: New frontiers: self-assembly and nanoelectronics. Computer **34**(1), 34–43 (2001)
9. Butts, M.; DeHon, A.: Molecular electronics: devices, systems and tools for Gigagate, Gigabit chips. In: In ICCAD-2002, pp. 433–440 (2002)
10. Strukov, D.B.; Likharev, K.K.: CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices. Nanotechnology **16**(6), 888–900 (2005)
11. Trel, O.; Lee, J.H.; Ma, X.; Likharev, K.K.: Neuromorphic architectures for nanoelectronic circuits. Int. J. Circuit Theory Appl. **32**(5), 277–302 (2004)
12. Goldstein, S.C.; Budiu, M.: NanoFabrics: spatial computing using molecular electronics. In: 28th Annual International Symposium on Computer Architecture, 2001. Proceedings, pp. 178–189 (2001)
13. DeHon, A.; Likharev, K.K.: Hybrid CMOS/nanoelectronic digital circuits: devices, architectures, and design automation. In: IEEE/ACM International Conference on Computer-Aided Design, 2005. ICCAD-2005, pp. 375–382 (2005)
14. Snider, G.S.; Kuekes, P.J.; Williams, R.S.: Crossbar nanocomputers. Sci. Am. **293**(5), 72–76 (2005)
15. Strukov, D.B.; Likharev, K.K.: CMOL FPGA circuits. In: In Proceedings of International Conference on Computer Design, CDES2006, pp. 213–219 (2006)
16. Stan, M.R.; Franzon, P.D.; Goldstein, S.C.; Lach, J.C.; Ziegler, M.M.: Molecular electronics: from devices and interconnect to circuits and architecture. Proc. IEEE **91**(11), 1940–1957 (2003)
17. Tahoori, M.B.; Mitra, S.: Fault detection and diagnosis techniques for molecular computing. In: In NanoTech (2004)
18. Brown, J.G.; Blanton, R.D.: CAEN-BIST: testing the nanofabric. In: Test Conference, 2004. Proceedings. ITC 2004. International, pp. 462–471 (2004)
19. Tehranipoor, M.; Rad, R.M.P.: Built-in self-test and recovery procedures for molecular electronics-based nanofabrics. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **26**(5), 943–958 (2007)
20. Joshi, M.V.; Al-Assadi, W.K.: A BIST approach for configurable nanofabric arrays. In: 8th IEEE Conference on Nanotechnology, 2008. NANO '08. pp. 695–698 (2008)
21. Sait, S.M.; Arafeh, A.M.: Cell assignment in hybrid CMOS/nanodevices architecture using Tabu Search. Appl. Intell. **40**(1), 1–12 (2013)
22. Strukov, D.B.; Likharev, K.K.: 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, pp. 131–140, New York, NY, USA. ACM (2006)
23. Hung, W.N.N.; Gao, C.; Song, X.; Hammerstrom, D.: Defect-tolerant CMOL cell assignment via satisfiability. Sensors J. IEEE **8**(6), 823–830 (2008)
24. Xia, Y.; Chu, Z.; Hung, W.N.N.; Wang, L.; Song, X.: CMOL cell assignment by genetic algorithm. In: 2010 8th IEEE International NEWCAS Conference (NEWCAS), pp. 25–28 (2010)
25. Chu, Z.; Xia, Y.; Hung, W.N.N.; Wang, L.; Song, X.: A memetic approach for nanoscale hybrid circuit cell mapping. In: 2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD), pp. 681–688 (2010)
26. Xia, Y.; Chu, Z.; Hung, W.; Wang, L.; Song, X.: An integrated optimization approach for nano-hybrid circuit cell mapping. IEEE Trans. Nanotechnol. **PP**(99), 1 (2011)
27. Xu, C.; Nepal, K.: Ant-colony-optimization based heuristic searching algorithm for cell assignment in a hybrid cmos/nano circuits (cmol) array. In: 2014 IEEE 14th International Conference on Nanotechnology (IEEE-NANO), pp. 262–267 (2014)
28. Sait, S.M.; Arafeh, A.M.: Efficient CMOL nanoscale hybrid circuit cell assignment using Simulated evolution heuristic. In: Proceedings of the great lakes symposium on VLSI, GLSVLSI '12, pp. 21–26, New York, NY, USA. ACM (2012)
29. Sait, S.M.; Arafeh, A.M.: Tabu Search based cells placement in nanofabric architectures with restricted connectivity. In: 2013 14th International Symposium on Quality Electronic Design (ISQED), pp. 487–493 (2013)
30. Yellambalase, Y.; Choi, M.: Cost-driven repair optimization of reconfigurable nanowire crossbar systems with clustered defects. J. Syst. Archit. **54**(8), 729–741 (2008)
31. Huang, J.; Tahoori, M.B.; Lombardi, F.: On the defect tolerance of nano-scale two-dimensional crossbars. In: 19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, 2004. DFT 2004. Proceedings, pp. 96–104 (2004)
32. Tahoori, M.B.: 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, pp. 668–672, Washington, DC, USA. IEEE Computer Society (2005)
33. Arafeh, A.M.; Sait, S.M.: Cells reconfiguration around defects in CMOS/nanofabric circuits using Simulated Evolution heuristic. In: 2015 16th International Symposium on Quality Electronic Design (ISQED) (2015)
34. Likharev, D.B.; Strukov, K.K.: CMOL: devices, circuits, and architectures. Introd. Mol. Electron. Lect. Notes Phys. **680**, 447–477 (2005)
35. Sait, S.M.; Youssef, H.: Iterative Computer Algorithms with Applications in Engineering: Solving Combinatorial Optimization Problems. IEEE Computer Society Press, California (1999)
36. Stapper, C.H.: Simulation of spatial fault distributions for integrated circuit yield estimations. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **8**(12), 1314–1318 (1989)

37. Stapper, C.H.: Simulation of spatial fault distributions for integrated circuit yield estimations. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **8**(12), 1314–1318 (1989)
38. Brglez, F.; Bryan, D.; Kozminski, K.: Combinational profiles of sequential benchmark circuits. In: IEEE International Symposium on Circuits and Systems, 1989, vol. 3, pp. 1929–1934 (1989)
39. Sentovich, E.M.; Singh, K.J.; Lavagno, L.; Moon, C.; Murgai, R.; Saldanha, A.; Savoj, H.; Stephan, P.R.; Brayton, R.K.; Vincentelli, A.S.: SIS: a system for sequential circuit synthesis. Electronics Research Laboratory Memorandum, (UCB/ERL M92/41) (1992)
40. Lee, H.K.; Ha, D.S.: HOPE: an efficient parallel fault simulator. In: Proceedings, 29th ACM/IEEE Design Automation Conference, 1992, pp. 336–340 (1992)