Tabu Search Based Cells Placement in Nanofabric Architectures with Restricted Connectivity

Sadiq M. Sait and Abdalrahman M. Arafeh
Department of Computer Engineering
Center for Communications and IT Research, Research Institute
King Fahd University of Petroleum & Minerals
Dhahran-31261, Saudi Arabia
email: {sadiq, arafeh}@kfupm.edu.sa

Abstract— New advances in nano-electronics have led to the introduction of CMOL (CMOS/Nano-devices hybrid) circuits which consists of an overlay of a nanowires over a CMOS stack. CMOL circuits can implement a netlist of NOR gates and Inverters using diode-like nanodevices. CMOL has inherently restricted connectivity due to limited nanowires length. Therefore connectivity of the circuit's elements is constrained to be within a certain radius, else an intermediary buffers are required.

In this paper we present a Tabu search (TS) algorithm to address cells placement problem in CMOL. The Heuristic is engineered to provide feasible circuit implementations by efficient exploration of search space. Empirical results for ISCAS'89 benchmarks are compared with previous solutions using GA, MA, and LRMA heuristics. Results show that in almost all cases, TS exhibits more intelligent search of the solutions subspace, and is able to find better solutions. For all tested benchmarks over 90% reduction in average CPU processing time when compared with best published techniques was obtained.

Keywords—CMOL, Tabu Search, Combinatorial Optimization, Search Heuristics, nanofabric, assignment, VLSI.

I. Introduction

Feature size scaling in CMOS technology has led to difficulties in manufacturing due to short channel effects, doping fluctuations and expensive lithography process. Meanwhile, advances in nanoelectronics are expected to achieve high density of devices and operate at THz frequencies [1]. Many effective applications have been proposed that use molecular nanodevices, nanowires, and nano-crossbar fabrics [2], [3]. A new trend is emerging for combining the flexibility and high fabrication yield advantages of CMOS technology with nanometer-scale molecular devices. A self-assembly of two-terminal nanodevices, with nanowire crossbar fabrics, enables high functional density and sustains acceptable fabrication costs. Likharev and Strukov [2] introduced a hybrid semiconductor/nanowire/molecular integrated circuit called CMOL, which uses two levels of perpendicular nanowires as crossbar interconnection on top of inverter-based CMOS stack, and showed possible applications of CMOL in field programmable gate arrays (FPGA) [4], neuromophic CrossNets [5], and in memories [6].

Assigning cells to slots is an important step in the process of electronic design automation. The assignment problem has been proven to be NP-hard. Overtime, the objective of placement has changed from reducing the overall wirelength to reduce the area, to improving timing performance, and then to reducing the overall power dissipation. With new advances in technologies come new issues. The above advancement that drastically improves the circuit density, CMOL, requires combinational logic to be implemented from a netlist of NOR gates and Inverters by programming nano-devices placed between overlapping nanowires. Like in the case of most programmable devices, the length of the nanowires is restricted, and therefore connectivity of the circuit elements is constrained to be within a certain radius, else additional buffers are required.

Recently, several proposals had been introduced for cell placement/assignment on FPGA-like CMOL architecture. Like in other nano-fabric crossbars and FPGA like devices, nanowires break at fixed intervals confining CMOL cell connectivity to a fixed number (M) of other cells located within its proximity square-like connectivity domain. Each CMOL cell must be connected to one of its proximity cell members, and failure to do so will require the insertion of a buffer which results in increase of congestion and delay. The problem here is to find an assignment that will result in smallest number of additional buffers.

Likharev et al utilized existing FPGA CAD tools to perform placement and routing on 4×4 tile-based version of CMOL [4], [7]. They used reserved routing cells and recursive routing algorithm for inter-tile routing. Hossein et al [8] proposed a recursive method for removing routing congestion by keeping and ranking placement solutions in final iterations of the placement algorithm according to cost. Subsequently, when routing of best placement configuration failed, another placement solution was considered until routing was satisfied. Instead of working at tiles level, Hung et al [9] encoded the CMOL cell assignment as a Satisfiability problem at cells level, where placement solution is found when all Boolean constraints are satisfied. However, when circuits sizes increased the computation time became exhibitant.

Previous attempts to use sub-optimal search heuristics are reported in [10], [11], [12]. Genetic Algorithm (GA) [10] were used with two dimensional block PMX crossover operator and mutation, where the fitness function evaluated the Manhattan distance between connected cells. Nonetheless,

memory requirements, choices of data structure for chromosomes representation, and computation time are significant disadvantages of GA. A more elaborate work was reported in [11]; where Memetic computing approach was used by implementing a hybrid of Genetic Algorithm and Simulated Annealing (SA) local-based search heuristic. SA was used in each generation to enhance offsprings which resulted from PMX crossovers and pairwise interchange mutations in GA. Hung et al [12] extended their work on Memetic approach by integrating self-learning operators using Lagrangian Multipliers (LRMA). Lagrangian relaxation technique (LRT) was applied in population goodness function by assigning Lagrangian multipliers to penalty values corresponding to problem constraints and repeatedly updating them. Results reported using LRMA approach are promising, however, more computations are needed for penalty updating mechanism.

The aim of this work is to investigate the cell assignment in CMOL nanofabric crossbar architecture. We'll be addressing the complexity associated with the confined CMOL nanowires crossbar on the logic connectivity and circuits implementation. The rest of the paper is arranged as follow: in the next section we provide a background about CMOL FPGA-like architecture. Section 3 details the problem formulation, Section 4 outlines Tabu Search, a heuristic engineered to solve our combinatorial optimization problem. Section 5 contains the empirical results, comparison and further discussion about the problem behavior. Finally, we conclude the paper and provide final remarks.

II. CMOL FPGA Architecture

CMOL cell-based, field-programmable gate array (FPGA)-like architecture is based on integrating conventional four-transistor MOSFET CMOS cell with uniform reconfigurable nanowire fabric. Each cell consists of CMOS inverter and two pass transistors. Two-terminal nanodevices "latching switches", that have two metastable internal states, are self-assembled at each crosspoint in CMOL fabric and provide diode-like I-V curves for logic circuits implementation. 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. That results in abundant available nanodevices that can serve both inter-cells connectivity and wiring-logic. CMOS stack is connected to nano-fabric by Metal pins that span to top and bottom nanowire levels as shown in Figure 1(a). Two CMOS inverters (i.e., inverter A and inverter C) are connected by pin-nanowire-nanodevice-nanowire-pin connection. The electrical representation of four inverter-based CMOS cells and corresponding nanowire and nanodevices is shown in Figure 1(b). Inverter A has two pins; pin1 connects the input of the CMOS inverter to one of the nanowires levels making the nanofabric, while pin2 connects the CMOS inverter's output to the second level of nanowires. The upper right cell (inverter A) is connected to the lower left cell (inverter C) by activating the appropriate nanodevice (nd1) in the crosspoint between the nanowire connected to output of inverter A and nanowire connected to input of inverter C. When two or more nanodevice on the same nanowire are activated as shown in Figure 1(b) (nd1 and nd2) the output of inverter C will be equivalent to NOR gate whose inputs are cell A and cell B. Wired-OR logic is implemented through nanowires and nanodevice.

CMOL nanowire crossbar is rotated by angle $\alpha = \arcsin(F_{nano}/\beta F_{CMOS})$ related to the CMOS pins that are arranged into a square array with side of $2\beta F_{CMOS}$ as shown in Figure 1(c), where F_{CMOS} is CMOS half-pitch, and β is a factor larger than 1. This approach allows a unique access to any nanodevice via the appropriate pin pair. Each CMOS cell has an area of $A = (2\beta F_{CMOS})^2$. Like other nano-fabric crossbars, CMOL's nanowires break at repeated intervals of $L = 2\beta^2 F_{CMOS}^2$ confining CMOL cells connectivity to only $M = 2r(r-1) - 1$ other cells located within its proximity square-like "Connectivity Domain" as shown in Figure 2, where r is an integer value that indicates the connectivity domain diameter and represents the constraint of CMOL placement. Each CMOL nanodevice is uniquely addressed by two pins from two different CMOL cells. During the configuration process an address decoder 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 set into ON (low-resistance) state or OFF (high-resistance). If the nanowires and nanodevices shown in Figure 3(b) are activated, the CMOL circuit will be equivalent to circuit shown in Figure 3(a). The first NOR gate of the circuit can be implemented by connecting inputs 'A' and 'B' with inverter '1' to satisfy both connectivity and logic wiring for the desired gate. The abundance of available nanodevices and nanowires provides a variety of different possible configurations for the implementation of one circuitry. Among those their could be only certain configurations that satisfy connectivity domain constraint and do not require additional routing resources.

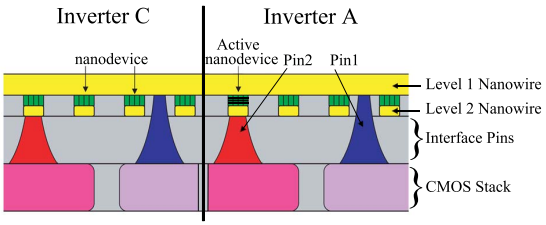
III. Problem Formulation

The placement or assignment of cells in order to minimize a cost function is an NP-hard problem [13]. Even one dimensional placement, the simplest possible, is hard to solve. In 2-D array of n locations there are as many as

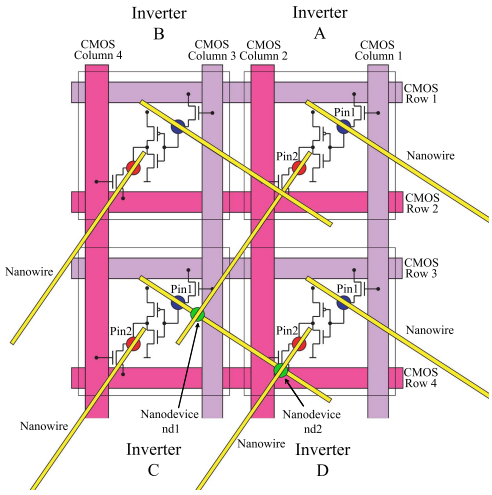
$$S = n(n-1)(n-2)...(n-m) \tag{1}$$

arrangements for placing m cells, where m could be in thousands. Overtime, heuristic techniques have been developed for solving the placement problem and to find a good solution in polynomial function of m.

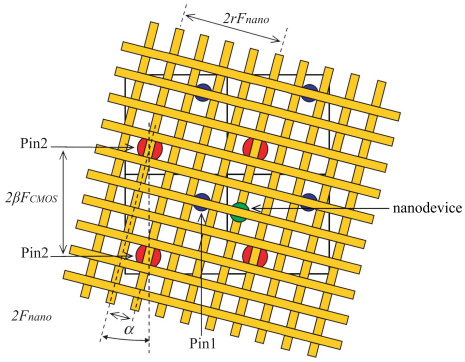
Given a collection of NOR/INV gates, and the collection of nets (the set of ports to be connected together), the CMOL placement problem consists of finding suitable locations for each gate under the constraint of connectivity domain and given a cost function. Formally the problem can be restated as: for a set of gates $G = g_1, g_2, g_3, ..., g_m$



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



(b)Electrical representation of four CMOL cells and corresponding nanowires.



(c)Nano-fabric inclined by α on top of four CMOS cells, only one nanodevice is shown.

Fig. 1. Low-level structure of CMOL circuit: the incline angle $\alpha \ll 1$ and dimensionless parameter β satisfy two conditions, $\sin\alpha = F_{nano}/\beta F_{CMOS}$ and $\cos\alpha = rF_{nano}/\beta F_{CMOS}$ where r is an integer.

and a set of netlists $\Gamma = \gamma_1, \gamma_2, \gamma_3, ..., \gamma_m$ where $\gamma_i = \{fan-in_i \ \& \ fan-out_i\}$ of g_i and given a set of slots or locations $L = L_1, L_2, L_3, ..., L_n$ where $m \leq n$, the placement problem is to assign each $g_i \in G$ to a unique location L_j such that the objective is optimized. Positions are defined by the coordinate values (x_j, y_j) and the subset of G that represent inputs/outputs may be pre-assigned fixed locations or constrained to certain positions.

Each CMOL cell can implement one inverter or one NOR gate with multiple fan-in, however, complying to connectivity constraint can be substantially harder if gates of high fan-in are allowed. Unlike conventional CMOS-based cell assignment, CMOL cell placement is constrained to *"Connectivity Domain"* of radius r. Each CMOL cell is connectable to one of its proximity cell members, any violation of this constraint would impose further processing (i.e., buffer insertion) to satisfy connectivity. However, such process would cause more congestion to the already congested CMOL circuit and could results in substantial increase of timing delay. Mathematically, the "Connectivity Domain" can be defined as follow. Given a gate and its netlist $(g_i, \gamma i)$ placed in location L_i, for any gate $g_k \subseteq G$ and g_k in the netlist γi the following inequality should be satisfied.

$$dist(L_j, L_k) \leq r \tag{2}$$

Where L_k is the location of g_k, *dist* is Manhattan distance, and r is CMOL connectivity diameter. The objective of CMOL cell assignment is to satisfy the constraint in Inequality 2, and to minimize distance between connected gates in circuit G. Failing to comply with CMOL constraint will result in an implementation that has more delay and area requirements. The complexity of CMOL placement arise from the overlap in connectivity domain of adjacent cells as shown in Figure 2, that results in fewer connectivity choices.

IV. Tabu Search

Tabu Search is a general iterative metaheuristic for solving combinatorial optimization problems. TS is a heuristic that proceeds by making iterative perturbations while preventing cycling to certain number of recently visited points in search space. The TS procedure starts from an initial feasible solution S (current solution) in the search space Ω. A neighborhood $\aleph(S)$ is defined for each S. A sample of neighbor solutions $\mathbf{V}^* \subset \aleph(S)$ is generated called *trial* solutions ($n = |\mathbf{V}^*| \ll |\aleph(S)|$), and comprises what is known as the candidate list. From this generated set of trial solutions, the best solution, say $S^* \in \mathbf{V}^*$ is chosen for consideration as the next solution. A solution $S^* \in \aleph(S)$ can be reached from S by an operation called a move to S^*. The move to S^* is considered even if S^* is worse than S, that is, $Cost(S^*) > Cost(S)$. Selecting the best move in $\mathbf{V}^*$ is based on the supposition that good moves are more likely to reach the optimal or near-optimal solutions. The best candidate solution $S^* \in \mathbf{V}^*$ *may* or *may not* improve the current solution, but is still considered. It is this feature that enables *escaping* from local optima. However, with this strategy, it is possible to reach the local optimum, since moves with $Cost(S^*) > Cost(S)$ are accepted, and then in a later iteration return back to local optimum.

In order to prevent returning to previously visited solutions a memory or list $\mathbf{T}$, known as *tabu list*, is maintained. This list contains information that to some extent forbids the search from returning to a previously visited solution. Whenever a move is accepted, its attributes are introduced into the tabu list $\mathbf{T}$. Move reversal are prevented for next

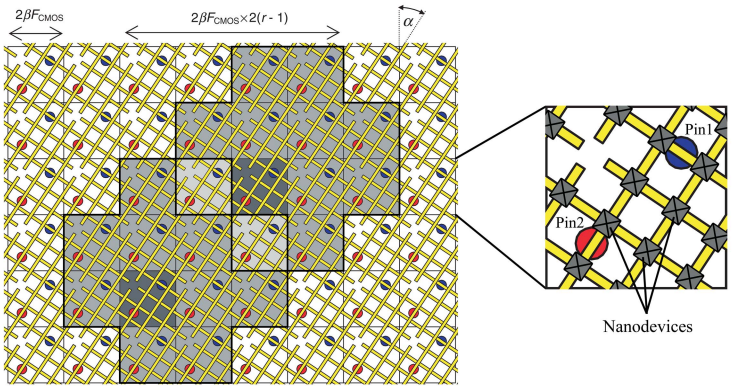


Fig. 2. CMOL FPGA topology: for $r = 3$, $M = 2r(r-1)-1 = 11$ cells in the "Connectivity Domain" (Highlighted by dark line) for the input pin of cells painted in dark-grey. The overlap between connectivity domain of two cells is shown in light grey.

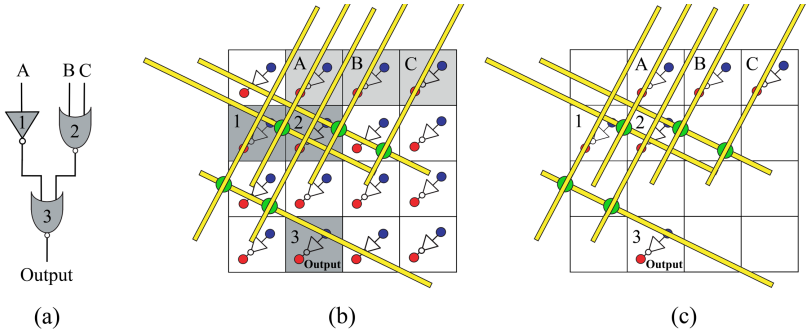


Fig. 3. Example of CMOL circuit: (a) NOR/INV logical circuit; (b) CMOL implmentaion of (a), (c) showing only used cells. Shaded cells are connected through combination of nanowires, nanodevices and CMOS pins.

$k = |\mathbf{T}|$ iterations because they *might* lead back to a previously visited solution. The tabu list can be visualized as a window on accepted moves; moves which tend to undo previous moves within this window are forbidden.

In some cases, it is necessary to overrule the tabu status since only move attributes (not complete solutions) are stored in tabu lists. These tabu moves may also prevent the consideration of some solutions which were not visited earlier. This is done with help of the notion of *aspiration criterion*. Aspiration criterion is a device used to override the tabu status of moves whenever appropriate. It temporarily overrides the tabu status if the move is sufficiently good. Aspiration criterion must make sure that the reverse of a recently made move leads the search to an unvisited solution, generally a better one [13].

One of the Tabu search algorithm parameters is the size of the tabu list. A small tabu list size is preferred for exploring the solution near a local optimum, and a larger tabu list size is preferable for breaking free of the vicinity of local minimum. The list size varying between 5 and 12 have been used in many applications. Any aspect (feature or component of a solution) that changes as a result of a move from S to S^{trial} can be an attribute of that move, where a single move can have several attributes. The duration for which a move containing the particular tabu attribute is forbidden (the size of tabu list) is called Tabu tenure. An algorithmic description of a simple implementation of the tabu search is given in Figure 4.

Ω : Set of feasible solutions (i.e., placements).
S : Current solution.
S^* : Best admissible solution.
$Cost$: Objective function (Reduce # of buffers).
$\aleph(S)$: Neighborhood of $S \in \Omega$.
$\mathbf{V}^*$: Sample of neighborhood solutions.
$\mathbf{T}$: Tabu list.
$\mathbf{AL}$: Aspiration Level.

Begin
Start with an initial feasible solution (placement) $S \in \Omega$.
Initialize tabu lists and aspiration level.
For fixed number of iterations **Do**
    Generate neighbor solutions $\mathbf{V}^* \subset \aleph(S)$.
    (Each solution results from the swap of two cells).
    Find best $S^* \in \mathbf{V}^*$.
    If move S to S^* is not in $\mathbf{T}$ **Then**
        Accept move & update best solution.
        Update tabu list (Store swap reversal).
        Update aspiration level.
        ($\mathbf{AL}$ = Cost of best solution seen so far).
        Increment iteration number.
    Else
        If $Cost(S^*) < \mathbf{AL}$ **Then**
            Accept move - update best solution.
            Update tabu list & aspiration level.
            Increment iteration number.
        EndIf
    EndIf
EndFor
End.

Fig. 4. Algorithmic description of short-term Tabu Search (TS).

A. Solution Representation and Initialization

A placement solution is an arrangement of logic cells in two dimensional layout surface. The representation used in this work is in the form of a 2-D grid. The layout is constructed by computing the number of required CMOL cells to fit each benchmark circuit. The outer cells of the grid are reserved for I/O pins, where I/O pins moves are restricted to these reserved locations. In the initialization phase each logic gate is assigned a positive integer value that distinguish it from the rest. Then, the encoded logic gates are randomly assigned in the 2-D layout.

B. Cost Evaluation

The main objective of placement is to find a feasible assignment of cells in which all connections are satisfied. Since, in CMOL all cells are connected via pre-assembled nanowires, the problem we are trying to optimize is to place connected cells within each others connectivity domain as to avoid insertions of additional buffers. Therefore, we should have a measure which can quantify the overall quality of the solution. A conventional approach is to calculate the number of nets that violate connectivity domain constraint. The overall cost of a solution is the total number of connectivity domain violating nets (the number of additional buffers that are needed to satisfy all connections). The cost of each gate $g \in G$ is expressed in Equation 3, where the overall circuit's cost is the sum of individual gates cost.

$$C_i = \sum_{j \in \gamma(i)} u_{i,j} \tag{3a}$$

$$u_{i,j} = \begin{cases} 1 & \text{if } dist_{i,j} > r \\ 0 & \text{otherwise} \end{cases} \tag{3b}$$

C. Neighborhood Solutions Generation

In each iteration we generate a number of neighbor solutions (i.e., candidate list) by making perturbations as follows: two cells (two I/O pins or two logic cells) are selected randomly, then their locations are interchanged. Each solution in the candidate list is evaluated based on the change in number of buffers before and after the swap. If two or more neighborhood solutions have equal swap cost, which also happens to be the best cost in the candidate list, the solution with lesser Manhattan distance is chosen. We have experimented with different sizes of candidate list; Figure 5 shows the final cost yielded by TS in four benchmark circuits when candidate list size is changed, given that all other parameters are constant. It is clearly seen that for this problem TS had better results when more neighbor solutions are considered. Candidate list size of 50 is reaching the optimal solution of zero buffers when $r = 12$, thus this size has been used throughout our implementation.

D. Tabu List and Aspiration Level

Different tabu attributes were tested, when two cells i and j are swapped. One attribute was to forbid moves related to cell i, that means any move included i even swapping i with j was tabued. Another experiment considered both i and j, forbidding any perturbations that include either of them. The Tabu attribute of a move that is used in all results reported in this paper is swap reversal. If two cells involved in interchange the reversal of this move is forbidden. A short term memory element is used throughout the implementation where experiments of tabu list size ranging from 5 to 12 were conducted. It was concluded that change in tabu list size in this range has little impact on the quality of the solutions, thus the size of tabu list is taken as a fixed value equal to 5. The aspiration criterion is based on the following: if the current solution is the best seen so far (i.e., better than the global best solution) then tabu restriction is overridden and the current solution is accepted as new best solution and tabu list is updated.

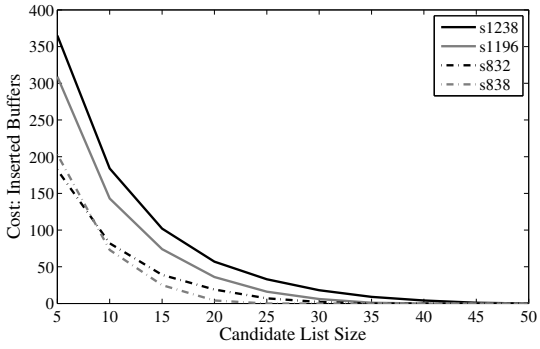


Fig. 5. Final cost yielded by TS in four circuits vs. Candidate List size ($r = 12$).

V. Experimental Results

Evaluation of search heuristic efficiency and behavior is conducted using ISCAS'89 benchmarks [14]. Further consideration should be given to ISCAS'89 by replacing sequential elements' inputs and outputs with POs and PIs respectively [9]. ISCAS'89 benchmarks used in this work are mapped to NOR-based gates with maximum of five inputs. Tabu Search has been implemented using Java programming language and executed on a machine comparable to the one used by other simulations published in literature, it has 1.5 GHz Intel Pentium M processor with 512MB memory. Figure 6 shows the correlation between the number of inserted buffers and Manhattan distance. It's clear that TS is accepting bad moves in order to reach better solutions in terms of inserted buffers, which are also better in terms of Manhattan distance.

Table I shows the number of cells (i.e., NOR/INV logic gates), inputs and outputs of benchmark circuits used; Area ($Tiles$) is the area used by CMOL FPGA CAD 1.0 tool [4], while Area ($Row \times Column$) is the area used in GA [10], MA [11], LRMA [12] and TS. The heuristic stops when all violations are removed or when reaching a predefined number of iterations. The median value of results obtained from 20 runs for each circuit is reported where each run uses different seeds for random numbers.

TABLE I

ISCAS'89 Benchmarks: showing the number of *Cells* to be placed including *Gates*, *Inputs* and *Outputs*. *Area* is the size of CMOL 2-D grid. *AU%* is the fraction of utilized cells in CMOL grid.

Circuits	**Cells**	**Gates**	**Inputs**	**Outputs**	**Area** ($Tiles$)	**Area** ($Row \times Column$)	**AU%** ($Tiles$)	**AU%**
s27	19	8	7	4	64 (2×2)	25 (5×5)	18.75	32.00
s208	136	109	18	9	$256(4 \times 4)$	$169(13 \times 13)$	48.05	64.50
s298	122	85	17	20	$256(4 \times 4)$	$144(12 \times 12)$	48.83	59.03
s344	180	130	24	26	$400(5 \times 5)$	$196(14 \times 14)$	43.50	66.33
s349	184	134	24	26	$400(5 \times 5)$	$196(14 \times 14)$	26.50	68.37
s382	175	124	24	27	$400(5 \times 5)$	$196(14 \times 14)$	43.25	63.27
s386	164	138	13	13	$400(5 \times 5)$	$196(14 \times 14)$	54.75	70.41
s400	188	137	24	27	$400(5 \times 5)$	$196(14 \times 14)$	47.25	69.90
s420	299	248	34	17	$400(5 \times 5)$	$361(19 \times 19)$	75.00	68.70
s444	187	136	24	27	$400(5 \times 5)$	$196(14 \times 14)$	52.50	69.39
s510	304	266	25	13	-	$361(19 \times 19)$	-	73.68
s526	273	222	24	27	$576(6 \times 6)$	$324(18 \times 18)$	57.12	68.52
s641	302	206	54	42	$576(6 \times 6)$	$676(26 \times 26)$	50.17	30.47
s713	321	225	54	42	-	$676(26 \times 26)$	-	33.28
s820	447	400	23	24	-	$529(23 \times 23)$	-	75.61
s832	454	407	23	24	-	$529(23 \times 23)$	-	76.94
s838	606	507	66	33	-	$676(26 \times 26)$	-	75.00
s1196	675	613	31	31	-	$729(27 \times 27)$	-	84.09
s1238	724	662	31	31	-	$784(28 \times 28)$	-	84.44

TABLE II

ISCAS'89 Comparison With CMOL CAD, GA, MA and LRMA - ($r = 12$).

Circuits	**CMOL CAD 1.0**		**GA [10]**			**MA [11]**			**LRMA [12]**			**Tabu Search**		
	Delay	Time	Delay	Time	Buf	Delay	Time	Buf	Delay	Time	Buf	Delay	Time	Buf
s27	9	1	7	0.01	0	7	0.01	0	7	0.01	0	7	0.01	0
s208	18	3	16	1.12	0	16	0.12	0	16	0.10	0	16	0.01	0
s298	13	7	11	0.17	0	11	0.11	0	11	0.09	0	11	0.01	0
s344	20	8	18	0.57	0	1	0.29	0	18	0.16	0	18	0.01	0
s349	20	7	18	0.49	0	18	0.28	0	18	0.18	0	18	0.01	0
s382	13	7	11	1.60	0	11	0.38	0	11	0.32	0	11	0.03	0
s386	16	11	10	1.05	0	10	0.33	0	10	0.34	0	10	0.03	0
s400	15	8	11	2.12	1	11	0.40	0	11	0.34	0	11	0.02	0
s420	20	8	16	8.50	1	16	3.41	0	16	1.57	0	16	0.07	0
s444	17	9	11	1.86	2	11	0.40	0	11	0.34	0	11	0.03	0
s510	-	-	18	16.56	2	18	7.56	0	18	3.42	0	18	0.18	0
s526	16	13	11	9.75	5	11	4.36	0	11	1.59	0	11	0.48	0
s641	25	8	23	82.66	15	**19**	**39.40**	**4**	16	22.02	0	**16**	**6.27**	**0**
s713	-	-	24	52.84	34	19	30.11	3	19	**41.77**	**2**	19	**8.69**	**0**
s820	-	-	15	77.52	41	12	61.71	10	12	**54.09**	**6**	12	**11.77**	**0**
s832	-	-	16	69.27	54	12	60.17	11	12	**63.77**	**4**	12	**10.55**	**0**
s838	-	-	28	201.37	50	24	85.62	7	24	**100.40**	**4**	24	**4.48**	**0**
s1196	-	-	30	234.88	84	23	208.15	19	**24**	**179.47**	**9**	**23**	**6.87**	**0**
s1238	-	-	37	268.92	121	28	267.34	31	26	**353.00**	**9**	26	**12.87**	**0**
Average	-	-	17	54.28	22	15	**40.53**	**4**	15	**43.31**	**2**	15	**3.28**	**0**

Delay: Logic Levels.
Time: Computation Time in Seconds.
Buf: Buffers Inserted.

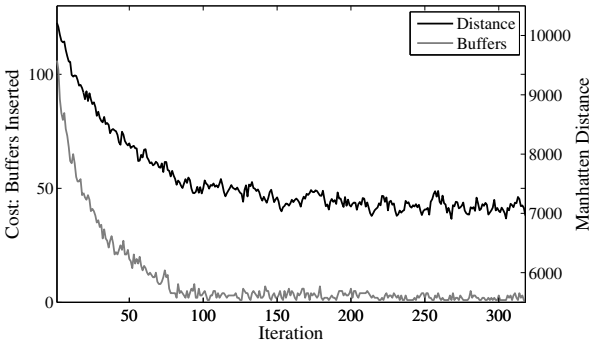


Fig. 6. Change of problem cost and Manhattan distance in TS iterations ($r = 12$ - s1238.blif).

A. Literature Comparison

Comparison is performed with CMOL FPGA CAD 1.0, we set the connectivity radius to $r = 12$. GA, MA and LRMA use population size equals to 24 and stopping criteria when fitness score is not updated for 50 times. The crossover rate in MA and LRMA is $RC = 0.33$ and mutation rate $RM = 0.01$. Simulated Annealing used in each of GA iterations has initial temperature $T = 0.2$ and terminating temperature 0.01. Table II shows the final results obtained for ISCAS'89 benchmarks when $r = 12$; ($Delay$) is the circuit's logical levels reported by SIS tool after inserting the buffers, computation time ($Time$) in seconds, (Buf) shows the number of inserted buffers to satisfy CMOL connectivity domain.

Tabu Search solutions are more effective than those of CMOL CAD 1.0 in terms of computation time, delay and

area utilization. The last two columns of Table I show that cell-based CMOL architecture has better area utilization $AU\%$ than that of tile-based architecture. Table II indicates that the tile-based approach is the most time consuming and the least effective in timing delay, it also fails to place big circuits.

Results obtained from implementation of TS for $r = 12$ are better than those obtained in GA, MA and LRMA in both computation time and Buffers count. TS required shorter CPU processing time due to its simplified operations compared to genetic crossover, mutation and Lagrangian multipliers calculation in LRMA. Table II shows that Tabu Search found the optimal solutions with zero buffers for all benchmarks, with 92% average computation time saving. For example, s1238 benchmark needed only 12.87 seconds in TS, comprising only a 3.6% of time needed by LRMA.

VI. **Conclusion**

In this paper we have shown how design automation algorithms, such as Tabu Search, can be employed for design problem of the emerging hybrid CMOS/nanodevices architectures, where cells connectivity is limited and nanodevices are used for logic implementation. We've analyzed the problem behavior and engineered a Tabu search solution that exploits better understanding of the limitations imposed by CMOL hybrid circuits' connectivity domain. Results obtained are better than those used in literature such as SA, GA and LRMA, with huge advantage in computations time saving. Further we are looking for the implementation of other search heuristics for CMOL cells placement and reconfiguration around defects problems.

VII. **Acknowledgements**

The authors acknowledge King Fahd University of Petroleum & Minerals for all support.

References

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

[2] 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.

[3] Gregory S. Snider and Stanley R. Williams. Nano/CMOS architectures using a field-programmable nanowire interconnect. *Nanotechnology*, 18(3):035204, 2007.

[4] 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.

[5] Konstantin K. Likharev. Crossnets: Neuromorphic hybrid CMOS/Nanoelectronic networks. *Science of Advanced Materials*, 3:322–332, 2011.

[6] Dmitri B. Strukov and Konstantin K. Likharev. Prospects for terabit-scale nanoelectronic memories. *Nanotechnology*, 16(1):137, 2005.

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

[8] Hossein Hamidipour, Parviz Keshavarzi, and Ali Naderi. Routing congestion removing of CMOL FPGA circuits by a recursive method. In *Proceedings of the 9th WSEAS international conference on Microelectronics, nanoelectronics, optoelectronics*, MINO'10, pages 75–79, Stevens Point, Wisconsin, USA, 2010. World Scientific and Engineering Academy and Society (WSEAS).

[9] 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.

[10] 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.

[11] 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.

[12] 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.

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

[14] F. Brglez, D. Bryan, and K. Kozminski. Combinational profiles of sequential benchmark circuits. In *Circuits and Systems, 1989., IEEE International Symposium on*, pages 1929 –1934 Vol.3, may 1989.