Front. Phys. 11(2), 116201 (2016)
DOI 10.1007/s11467-015-0531-8

RESEARCH ARTICLE

Efficient design method for cell allocation in hybrid CMOS/nanodevices using a cultural algorithm with chaotic behavior

Zhong-Liang Pan[1,†], Ling Chen[1], Guang-Zhao Zhang[2]

[1] *School of Physics and Telecommunications Engineering, South China Normal University, Guangzhou 510006, China*
[2] *Department of Electronics and Communications, Sun Yat-sen University, Guangzhou 510275, China*
Corresponding author. E-mail: †panscnu@126.com
Received August 13, 2015; accepted September 16, 2015

The hybrid CMOS molecular (CMOL) circuit, which combines complementary metal–oxide–semiconductor (CMOS) components with nanoscale wires and switches, can exhibit significantly improved performance. In CMOL circuits, the nanodevices, which are called cells, should be placed appropriately and are connected by nanowires. The cells should be connected such that they follow the shortest path. This paper presents an efficient method of cell allocation in CMOL circuits with the hybrid CMOS/nanodevice structure; the method is based on a cultural algorithm with chaotic behavior. The optimal model of cell allocation is derived, and the coding of an individual representing a cell allocation is described. Then the cultural algorithm with chaotic behavior is designed to solve the optimal model. The cultural algorithm consists of a population space, a belief space, and a protocol that describes how knowledge is exchanged between the population and belief spaces. In this paper, the evolutionary processes of the population space employ a genetic algorithm in which three populations undergo parallel evolution. The evolutionary processes of the belief space use a chaotic ant colony algorithm. Extensive experiments on cell allocation in benchmark circuits showed that a low area usage can be obtained using the proposed method, and the computation time can be reduced greatly compared to that of a conventional genetic algorithm.

Keywords nanodevices, structure design, cell allocation, CMOS technology, cultural algorithms

PACS numbers 62.23.st, 68.35.bg

1 Introduction

In recent years, the scaling of complementary metal–oxide–semiconductor (CMOS) technology has continued to progress, but it might not be cost-effective to use CMOS technology to manufacture devices scaled down to a few nanometers [1–3]. Therefore, molecular devices or nanodevices are being intensively investigated.

It is believed that the growing problems with the scaling of CMOS technology may be overcome by using the CMOS molecular (CMOL) circuit, which is a hybrid of CMOS circuits and nanodevices. The design and development of semiconductor/nanodevice hybrids is expected to play a role in managing the trade-off between nanodevice functionality and simplicity. The basic principle of CMOL hybrid circuits is to combine CMOS technology with molecular-scale nanodevices. One approach to realizing the CMOL circuit is to complement the CMOS stack with a few layers of added nanoelectronics in the form of a nanowire crossbar [4–6]. To tackle the problem of the interface between the CMOS stack and the nanodevices, in the CMOL circuit, the interface is provided with many sharp-tipped pins, which are on top of the CMOS stack and distributed over the entire circuit region.

In research on CMOL circuit design, Dong *et al.* [7] discussed the structure of CMOL circuits and presented new CMOL building blocks using transmission gates to obtain efficient combinational and sequential logics for the design of CMOL circuits. Strukov and Likharev [8] investigated a CMOL structure with two-terminal nanodevices; the structure combines a semiconductor–transistor CMOS stack and two levels of parallel nanowires, where the molecular-scale nanodevices are formed between the nanowires at every crosspoint. Abid *et al.* [9] discussed

Frontiers of Physics

3D CMOS nanohybrid technology, which can increase the density of the nanodevices and achieve higher performance than 2D CMOL technology, and performed architecture optimization, 3D integration, defect tolerance, and performance evaluation for a 3D CMOL field-programmable gate array (FPGA).

In terms of applications of CMOL circuits, Zaveri and Hammerstrom [10] investigated the hardware implementation of the Bayesian inference framework using hybrid nanotechnology in the form of a CMOS/nanowire/molecular hybrid. The results showed that CMOL was a promising candidate for implementing Bayesian inference, and it can exploit the very high-density storage and computation benefits of nanoscale technologies.

Afifi *et al.* [11] discussed the implementation of spiking neural networks on a CMOL architecture; they presented a spiking neural topology with spike-timing-dependent learning ability and described its building blocks, which can be easily mapped onto a CMOL architecture. Abid *et al.* [12] presented a design approach to CMOL circuits for efficient implementation of the advanced encryption standard (AES) cryptographic algorithm. The design approach can greatly reduce the power dissipation, area, and time delay compared to those of existing implementations of AES cryptographic systems.

For the cell assignment of CMOL circuits, Chen *et al.* [13] investigated whether a combinatorial circuit can be placed in a CMOL circuit and presented two theoretical results, which showed that when a reasonable connection domain size was given, any combinatorial circuit can be transformed to an equivalent circuit that can be placed in a CMOL FPGA circuit. Hung *et al.* [14] investigated defect-tolerant CMOL cell assignment using a satisfiability approach. They first transformed a logic circuit based on AND/OR/NOT gates to one based on NOR gates; second, they mapped the NOR gates to CMOL cells. The CMOL cell assignment was translated to Boolean conditions and then solved by the satisfiability approach. In addition, many types of static defects in the CMOL circuit architecture were also investigated.

Chu *et al.* [15] discussed the cell mapping of CMOL circuits under the structural connectivity domain constraint and proposed a memetic computing algorithm for cell assignment. The algorithm can take advantage of both the local search in simulated annealing and the manipulation strategy based on populations. In addition, they also proposed an approach to CMOL cell assignment based on dynamic interchange [16]. In this approach, the CMOL cell resources are first allocated and the gates to the CMOL cells are randomly mapped; then the positions of the gates are adjusted by interchanging and inserting buffers. Wang and Wang [17] investigated CMOL cell assignment and presented a pseudo-Boolean programming approach in which the CMOL cell assignment problem is encoded as a pseudo-Boolean constraint that is solved by a pseudo-Boolean solver.

In this paper, a new method is presented for cell allocation in CMOL circuits using a cultural algorithm with chaotic behavior.

2 Structure of CMOL circuits

Because of the parameter sensitivity, signal integration, and high fabrication costs, there are some difficulties with the application of current very-large-scale integration design and production technology to the nanometer or smaller regime [18–20]. Therefore, molecular devices or nanodevices are investigated using chemically directed self-assembly.

Nanodevices are ultrasmall and may operate with much lower power consumption than conventional microelectronic devices. The wiring and mass manufacturing techniques are two key factors to be addressed for practical application of nanodevices. One promising technology is the hybrid CMOL structure, which consists of the CMOS circuits, nanowires, and nanodevices arranged in several layers. First, the Boolean logic circuits can be realized with CMOL structure, for instance, a NOR gate can be assembled with three CMOS cells which consist of a single CMOS inverter and two nanodevices. Second, the memories can be built by using the CMOL circuits since the single latching switch can store a single bit of information. The CMOL memories may be as fast as the conventional CMOS memories, but they are much denser.

Figure 1 shows a cross section of a CMOL circuit. The CMOS circuit layer at the bottom includes traditional CMOS circuit components such as metal lines, gates, and function blocks.

In the CMOS layer, interface pins are used to connect the nanowires. The CMOL circuit also has two parallel

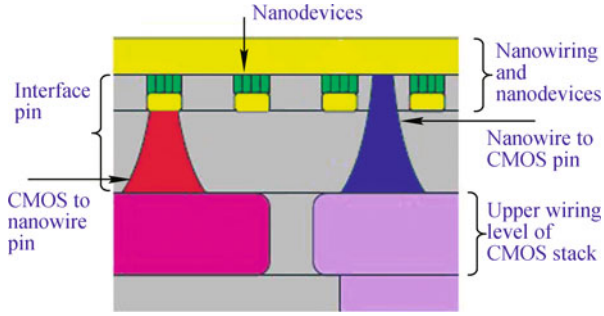

Fig. 1 The cross section of CMOL circuit.

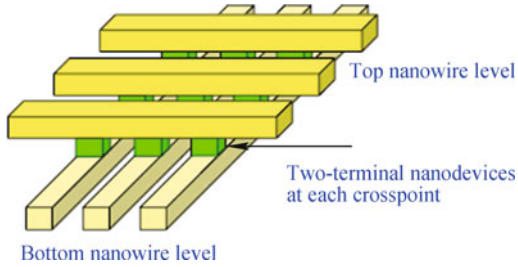

Fig. 2 The nanodevices are embedded at every cross point.

layers of nanowires at the top. Between the two nanowires layers, the nanodevices are embedded at every cross point of the nanowires, as shown in Fig. 2.

Figure 3 shows a top view of an example of a CMOL circuit in which every nanodevice can be addressed using two nanowires and their corresponding interface pins. For example, nanodevice 2 can be addressed by pins 1 and 3.

The nanodevices at each cross point of the nanowires can be multiterminal or two-terminal devices. Multiterminal devices have complex functions. Simple two-terminal devices are binary latching switches with two metastable states, ON and OFF. Figure 4 shows the schematic I–V curves of a two-terminal device.

If a voltage is applied to its terminals, the device can be in either the low-resistive ON state or the high-resistive OFF state. At a low voltage or if a negligible current flows through the device, it behaves as a typical resistor or diode, and it may be in the OFF state. A high voltage (greater than V_+ or smaller than V_-) can be used to switch this device between the ON and OFF states.

One effective approach to the implementation of logic circuits using the CMOL hybrid structure is based on reconfigurable regular structures such as FPGAs.

In this approach, an elementary CMOS cell consisting of two pass transistors and an inverter is connected to the nanowire/molecular subsystem using two pins. During configuration, the inverters are turned off, and the pass transistors can be used to set the binary state of each molecular device. Every pin of a CMOS cell may be connected using a nanowire–nanodevice–nanowire link to each of $2\mu^2 - 2\mu - 1$ other cells within a square connectivity domain around the pin, where μ is the radius of the connectivity domain. Figure 5 shows the general structure of the CMOL FPGA circuit; the connectivity domain appears light gray.

Figure 6 illustrates the implementation of a fan-in-two NOR gate with two inputs, A and B, and an output F.

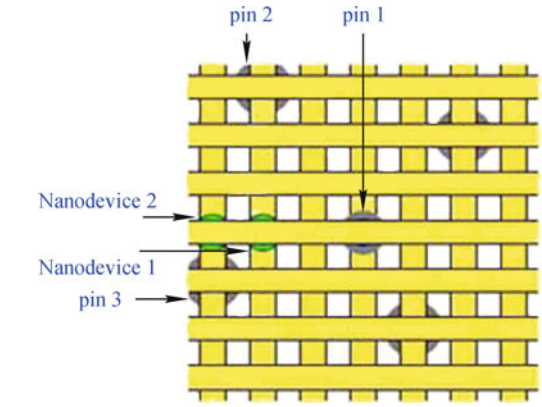

Fig. 3 The top view for an example of CMOL circuit.

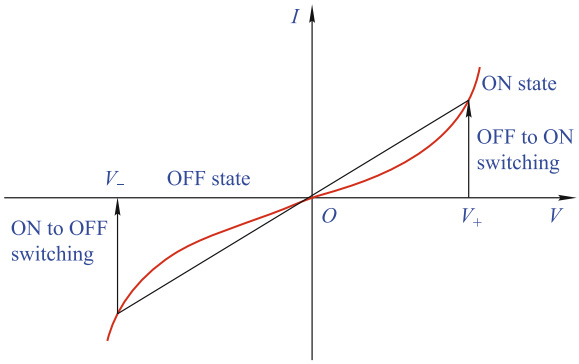

Fig. 4 The I–V features of two terminal devices.

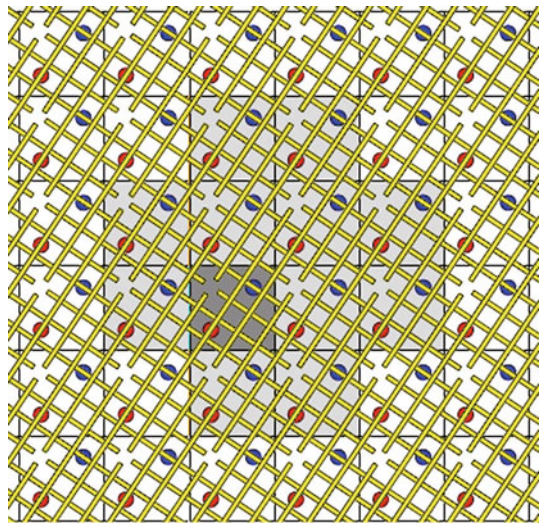

Fig. 5 The general structure of CMOL FPGA circuit.

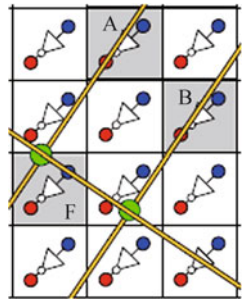

Fig. 6 The implementation for a fan-in-two NOR gate.

Similarly, the CMOL FPGA circuit can also be realized for gates with a larger fan-in and fan-out. Therefore, any logic function can be implemented using the CMOL FPGA.

3 Formulation of cell allocation problem

In the traditional circuit design procedure, a Boolean logic circuit is designed in accordance with the logic functionality being realized. Here, we consider the CMOL circuit to realize the functionality of conventional Boolean circuits.

First, the circuit logic functionality is transformed into a netlist of NOR gates. Second, each NOR gate is allocated and is mapped to a cell of the CMOL circuit, where a cell can be connected to only a limited number of neighboring cells.

Let G be the set of NOR gates in the given circuit netlist and C be the set of cells in the CMOL circuit. The cell allocations can be formulated as follows.

Given sets G and C, find a mapping f: G $\rightarrow$ C such that (1) $\forall\ k, j \in$G, $d(f(k), f(j)) \leqslant \mu$, (2) $f(i) \neq f(j)$ when $i \neq j$, (3) and the Manhattan distances of cells in the CMOL circuit are minimized.

If each NOR gate is assigned only to a particular cell, and the $f(k)$ is the position of the gate k (i.e., its X and Y coordinates) in the CMOL array, $d(f(k), f(j))$ indicates the Manhattan distance of $f(k)$ and $f(j)$. Further, the μ is the CMOL radius of the connectable domain and shows that each cell is connectable to one of its nearby cell members.

4 Cell allocation by cultural algorithm with chaotic behavior

In this section, we develop an algorithm for cell allocation in CMOL circuits that is based on the principles of cultural algorithms. One advantage of cultural algorithms, which add domain knowledge to evolutionary computation, is that the overall computational cost can be decreased [21].

The cultural algorithm consists of a population space, a belief space, and a communication protocol. The communication protocol describes how knowledge is exchanged between the population and belief spaces. The population space consists of many possible solutions for a given problem. The belief space serves as an information repository; individuals in the belief space store their experiences for other individuals to use [22,23].

The communication protocol gives the types of information to be exchanged between the population and belief spaces, indicates the rules that the individuals in population space can contribute to belief space by using the accept operation, and gives the way that the individuals in belief space can influence the new individuals in population space by using the influence operation.

In the implementation of a cultural algorithm for cell allocation, several features should be considered, such as the encoding mode of individuals and the realization of the population and belief spaces. In the following, these features will be investigated in detail.

The procedure of the entire cultural algorithm for cell allocation consists of the following steps.

Algorithm 1

Step 1. Initialize the population and belief spaces; i.e., generate initial individuals in the two spaces.

Step 2. Evaluate the individuals in the population and belief spaces; i.e., compute the fitness of every individual in the two spaces.

Step 3. For the individuals in the population space, perform the genetic algorithm with three populations in parallel competitive evolution, thus producing new individuals. Next, evaluate these new individuals and replace some old individuals with new individuals when the new ones are better.

Step 4. For the individuals in the belief space, perform the chaotic ant colony algorithm and produce new individuals. Next, evaluate these new individuals.

Step 5. Perform the accept operation and influence operation between the population and belief spaces to update the two spaces.

Step 6. Build the belief knowledge in the belief space.

Step 7. If the termination condition is satisfied, stop the algorithm; otherwise, go to Step 3.

In Algorithm 1, an individual is used to represent one scheme of cell allocation, i.e., one mapping result. Suppose the total number of NOR gates in the logic circuit is L, and the number of cells in the CMOL cell array is n. Then an individual is represented by an n-dimensional vector (z_1, z_2, $\cdots$, z_n). The value of component z_i belongs to 0, 1, 2, $\cdots$, L. If $z_i = 0$, then z_i represents that the i-th cell in the CMOL array is not occupied by any NOR gate. If $z_i = k$, where k belongs to 1, 2, $\cdots$, L, then z_i represents that the k-th cell in the CMOL array is occupied by the k-th NOR gate. For example, as shown in Fig. 7, where $L = 16$ and $n = 12$, the coding of the individual Z is Z = (2 5 0 7 11 0 8 0 9 4 1 12 3 10 0 6), and there are four unoccupied cells.

For an individual Z, if all the Manhattan distances between two gates are less than μ, which is the CMOL radius of the connectable domain, the fitness h of an

2	5	0	7
11	0	8	0
9	4	1	12
3	10	0	6

Fig. 7 The coding of the individuals.

individual Z is defined by Eq. (1) below; otherwise, the fitness h is defined by Eq. (2) below, where T is the total number of individuals in the population.

$$h = \frac{1}{\sum_{i=1}^{T}\sum_{j=1}^{T} d(i,j) + \beta_1}, \tag{1}$$

$$h = \frac{1}{(1+\beta_2)\cdot\sum_{i=1}^{T}\sum_{j=1}^{T} d(i,j)}, \tag{2}$$

where $d(i,j)$ represents the Manhattan distance between NOR gates i and j. Further, β_1 and β_2 are constants, for example, $\beta_1 = 0.001$, $\beta_2 = 0.01$.

The details of the implementations of Algorithm 1, such as the design of the belief space and population space, the accept operation, and the influence operation, are as follows.

(i) Realization of the population space

To realize the population space in Algorithm 1, we use the genetic algorithm with three populations in parallel competitive evolution. The detailed procedure is as follows.

Algorithm 2

Step 1. Set the parameter $k = 0$; produce an initial population with $3M$ individuals randomly.

Step 2. Divide the current population into three subpopulations $U(k)$, $V(k)$, and $W(k)$, where the number of individuals in each subpopulation is M.

Step 3. Apply the selection, crossover, and mutation operations to $U(k)$, $V(k)$, and $W(k)$, respectively. Repeat this step ξ times, where ξ is a positive integer.

Step 4. Migrate q_1 individuals from subpopulation $U(k)$ to $V(k)$. Migrate q_2 individuals from subpopulation $V(k)$ to $W(k)$. Migrate q_3 individuals from subpopulation $W(k)$ to $U(k)$.

Step 5. Remove one individual randomly from the current population. Add the best individual in the previous population to the current population, such that the best individual in each population always survives to the next population.

Step 6. If k is smaller than the given integer, set $k := k + 1$ and go to Step 2; otherwise, stop the algorithm.

In Algorithm 2, M is a positive integer, and the number of individuals in the population is $3M$. In Step 4, q_1, q_2, and q_3 are positive integers that are much smaller than M. The evolutionary strategies used in subpopulations $U(k)$ and $W(k)$ are identical; they apply genetic operations used in conventional genetic algorithms (CGAs). The evolutionary strategy used in subpopulation $V(k)$ is a niche technique based on a sharing mechanism.

The fitness function for subpopulation $V(k)$ is adjusted as follows. First, compute the fitness $c(s_i)$ of each individual s_i in subpopulation $V(k)$. Second, with regard to two arbitrary individuals s_i and s_j in $V(k)$, compute the number of components for which their values are not equal to each other; suppose it is u_{ij}. If $u_{ij} = 0$, set the parameter $g(u_{ij}) = 1$. If $u_{ij} > \lambda$, set $g(u_{ij}) = 0$ (where λ is a positive integer, $0 \leqslant \lambda \leqslant n$). If $0 < u_{ij} \leqslant \lambda$, set $g(u_{ij}) = 1 - u_{ij}/\lambda$. Third, compute

$$\theta(s_i) = \frac{c(s_i)}{\sum_{j=1}^{M} g(u_{ij})},$$

Then, $\theta(s_i)$ is the new fitness of individual s_i in subpopulation $V(k)$.

In Step 4 of Algorithm 2, individuals are migrated between two subpopulations as follows. Choose q_1, q_2, and q_3 individuals from subpopulations $U(k)$, $V(k)$, and $W(k)$, respectively, where the fitness of each individual being chosen is greater than the average fitness of the respective subpopulation. Concurrently, execute the following three substitutions: use the q_1 individuals chosen in $U(k)$ to replace q_1 individuals in $V(k)$, use the q_2 individuals chosen in $V(k)$ to replace q_2 individuals in $W(k)$, and use the q_3 individuals chosen in $W(k)$ to replace q_3 individuals in $U(k)$. Here, every individual being replaced is selected at random from the respective subpopulation, and its fitness is less than the average fitness of the respective subpopulation.

(ii) Realization of the belief space

To realize the belief space in Algorithm 1, we use the chaotic ant colony algorithm, which employs a chaotic approach to find better solutions whenever all the ants have finished their one operation.

The ant colony algorithm is a heuristic optimization technique that tries to mimic the behavior of ants in the real world. Real ants can choose the shortest path between their nest and a food resource when alterna-

tive paths between the two exist. When ants walk, they deposit a chemical substance called a pheromone along the way. If the ants arrive at a decision position, they make a probabilistic choice based on the intensity of the pheromone they smell. Ant colony algorithms attempt somehow to apply behavior similar to that of real ants to solve real-life problems.

To enhance the computational performance of the ant colony algorithm, for instance, to avoid the algorithm search becoming trapped in a local optimum, we use a chaotic strategy to find a better solution whenever all the ants have finished their one operation. We use the following chaotic system, which is defined by the chaotic mapping

$$z_{k+1} = \sin(2/z_k), \quad z_k \in (0,1), k = 0,1,2,\cdots. \tag{3}$$

The chaotic mapping in Eq. (3) has several features, such as stochasticity and sensitive dependence on the initial conditions. The evolution of individuals in the belief space using the chaotic ant colony algorithm proceeds as follows, where an individual in the belief space is treated as an ant.

Algorithm 3

Step 1. Set the initial value of the number of iterations B to 0.

Step 2. Generate the initial position of each ant using the chaotic system given by Eq. (3).

Step 3. Move each ant as follows. There are eight cells around ant k. For ant k, the probability of movement from cell j to cell i is given by

$$p_{ij}^k(t) = \frac{W(\sigma_i)\cdot R(\Delta_i)}{\sum\limits_{t\in D(j)} W(\sigma_j)\cdot R(\Delta_j)},$$

where $R(\Delta_i)$ is a weighting factor, and Δ_i is the magnitude of the difference between the current orientation and the previous direction in which the ant last moved. $D(j)$ indicates the sum over cell j, which is in the local neighborhood of k. The $W(\sigma_i)$ is a pheromone weighting function that measures the relative probabilities of moving to a cell with pheromone density σ_i. $W(\sigma_i)$ is given by

$$W(\sigma_i) = \left(1 + \frac{\sigma_i}{1 + u\cdot\sigma_i}\right)^s,$$

where s is related to the osmotropotaxic sensitivity, which is used to control the degree of randomness with which the ant follows the gradient of the pheromone. With regard to u, $1/u$ is the sensory capacity; this notation demonstrates that the ability of an ant to sense the pheromone will decrease somewhat at high concentrations.

Step 4. Compute the fitness of each ant.

Step 5. Perform pheromone deposition. For every ant k, pheromone deposition proceeds as follows. Adjust the pheromone τ_{ij} so as to make connections belonging to short tours more preferable:

$$\tau_{ij}^k(t+1) = (1-\rho)\cdot\tau_{ij}^k(t) + \Delta\tau_{ij}^k(t),$$

where ρ is the pheromone evaporation factor, $0 < \rho \leqslant 1$. If the connection between positions i and j is used by ant k, then $\Delta\tau_{ij}^k(t) = 1/L_k$; otherwise, $\Delta\tau_{ij}^k(t) = 0$, where L_k is the length of the tour built by ant k.

Step 6. Increase the value of B by 1; i.e., $B := B + 1$.

Step 7. If the value of B is smaller than the given integer, go to the Step 3; otherwise, stop the algorithm.

(iii) Realization of the accept and influence operations

The accept operation provides the rules by which individuals in the population space can contribute to the belief space; it also determines which individuals and behaviors may affect the knowledge in the belief space. In this paper, we adopt the following mode: individuals with minimal fitness in the belief space are replaced by the individual with maximal fitness in the population space when the individuals in the population space have evolved D_1 generations. D_1 is a given number of generations, for example, $D_1 = 8$.

The influence operation provides the rules by which individuals in the belief space can influence new individuals in the population space. In this paper, we adopt the following mode: when the individuals in the population space have evolved D_2 generations, we use a number of individuals in the belief space to replace the same number of individuals with lower fitness in the population space. D_2 is defined as $D_2 = D_0 + E_0\cdot(N_{max} - N_C)/N_{max}$, where N_{max} is the maximal number of evolution generations, N_C is the current number of evolution generations, and D_0 and E_0 are positive constants. Here, we select $D_0 = 5$ and $E_0 = 80$. Using the equation for D_2, we find that the belief space has little effect on the population space in the initial stage of the entire algorithm, and the belief space may have an increasingly strong effect on the population space as the algorithm proceeds. Thus, the search range of the entire algorithm can be enlarged gradually.

To summarize the discussion, in Section 4, we described in detail the implementation steps for cell allocation by the cultural algorithm. In Section 5, we present many experimental results.

5 Experimental results

Many experiments were conducted to examine cell allocation in CMOL circuits using the method described in this paper. The software programs were implemented in C++, and we ran them on a 3.06 GHz personal computer with 2 GB of main memory. We performed many experiments for the ISCAS89 benchmark circuits.

The benchmark circuits are described on the gate level, they can be implemented by using the partial scan or full scan structure. The sequential elements in these circuits are the D-type flip-flops with a single data input and a single uncomplemented output. The flip-flops can be expanded to functional level with a single clock or switch level with scan inputs by means of inserting additional reset and control signals.

Table 1 shows the features of the sequential benchmark circuits.

In Table 1, the column "Circuit" lists the names of the benchmark circuits. The columns "Primary input" and "Primary output" denote the number of primary inputs and primary outputs in a benchmark circuit, respectively. The column "Flip-flop" gives the total number of flip-flops in a benchmark circuit. The column "Gate" gives the total number of gates in a benchmark circuit.

These experiments on cell allocation in CMOL circuits were conducted as follows.

Step 1. For each sequential circuit in the ISCAS89 benchmark circuits, convert all the inputs and outputs of flip-flops (i.e., all the sequential elements) to primary outputs and inputs, respectively, such that each sequential circuit is transformed a combinational logic circuit.

Step 2. Use the logic synthesis tool SIS [24] to transform the combinational circuit obtained in Step 1 to a netlist of NOR gates.

Step 3. Perform the cell allocations.

For each NOR gate in the circuit netlist, allocate and map it to a cell of the CMOL circuit. Here the CMOL radius of the connectable domain is set to 12; i.e., $\mu = 12$.

The crossover and mutation operations in Algorithm 2 were implemented as follows. For the crossover operation, first, two parent individuals are chosen; second, two cut positions are selected randomly. Because simple exchange of parts between the cut positions may generate invalid solutions, the offspring are produced as follows: the part between the cut positions is selected from one parent while preserving the position and order of as many variables as possible from the second parent.

For instance, let the parent individuals be Z_1 = (3 5 0 0 4 6 2 1 0), Z_2 = (2 0 4 1 5 0 0 3 6). When the fifth and eighth positions are cut, if we perform a simple exchange, i.e., replacing the "4" and "1" in Z_1 with the "5" and "3" in Z_2, respectively, the offspring produced are (3 5 0 0 5 6 2 3 0) and (2 0 4 1 4 0 0 1 6), which are invalid. Valid offspring of Z_1 and Z_2 should be (1 4 0 0 5 6 2 3 0) and (2 0 5 3 4 0 0 1 6).

For the mutation operation, first, two component positions of a parent individual are selected randomly; second, the values at these two positions are exchanged. For example, for the individual Z_3 = (4 1 0 3 6 0 2 5 0), if the two selected component positions are the fourth and seventh, then the offspring (4 1 0 2 6 0 3 5 0) is produced.

In these experiments, the following parameters are used in the cultural algorithm: the maximal number of evolution generations is set to 1600, and the population sizes of the population and belief spaces are 40 and 20, respectively. The parameters in the crossover and mutation operations in Algorithm 2 for the evolution of individuals in the population space are as follows: the crossover rate is 0.96, and the mutation rate is 0.002.

In addition, to compare the experimental results, we also conducted experiments using the CAD software tool [25] and the conventional genetic algorithm (CGA) for cell allocation in CMOL circuits. Table 2 shows the experimental results obtained using the CAD tool; Table 3 compares the experimental results obtained using the the CGA and the method proposed in this paper.

Table 1 The features of the ISCAS89 benchmark circuits.

Circuit	Primary input	Primary output	Flip-flop	Gate
S208	11	2	8	96
S298	3	6	14	119
S349	9	11	15	161
S386	7	7	6	159
S400	3	6	21	162
S444	3	6	21	181
S713	35	23	19	393
S832	18	19	5	287
S1196	14	14	18	529
S1238	14	14	18	508

Table 2 The experimental results by using the CAD software tool.

Circuit	Area (Tiles)	Cells	AU (%)	Delay	Buffers	Time (s)
S208	256(4 × 4)	123	48.05	18	0	2.21
S298	256(4 × 4)	125	48.83	13	0	4.56
S349	400(5 × 5)	186	46.50	20	0	4.41
S386	400(5 × 5)	219	54.75	16	0	7.52
S400	400(5 × 5)	189	47.25	15	0	5.03
S444	400(5 × 5)	210	52.50	17	0	5.80
S713	–	–	–	–	0	–
S832	–	–	–	–	0	–
S1196	–	–	–	–	0	–
S1238	–	–	–	–	0	–

Table 3 The experimental results by using the CGA and the method in this paper.

Circuit	Area (Row×column)	Cells	AU (%)	CGA			This method in this paper		
				Delay	Buffers	Time (s)	Delay	Buffers	Time (s)
S208	169(13 × 13)	109	64.50	16	0	0.76	16	0	0.49
S298	144(12 × 12)	85	59.03	11	0	0.15	11	0	0.10
S349	196(14 × 14)	134	68.37	18	0	0.36	18	0	0.23
S386	196(14 × 14)	138	70.41	10	0	0.62	10	0	0.42
S400	196(14 × 14)	137	69.90	11	1	1.54	9	0	0.93
S444	196(14 × 14)	136	69.39	11	2	1.31	8	0	0.81
S713	676(26 × 26)	225	33.28	24	34	36.92	19	25	25.40
S832	529(23 × 23)	407	76.94	16	54	51.04	12	46	33.18
S1196	729(27 × 27)	613	84.09	30	84	175.32	24	63	116.40
S1238	784(28 × 28)	662	84.44	37	121	205.17	26	102	135.62

The following parameters are used for the crossover and mutation operations in the CGA: the maximal number of evolution generations is set to 1600, the population size is 60, the crossover rate is 0.96, and the mutation rate is 0.002.

In Tables 2 and 3, the column "Area" indicates the scale of the CMOL cell array territory. In particular, the "Tiles" column in Table 2 is the region generated on the basis of the logic structure of the circuit netlist, where the area of one "tile" is equal to the area of 16 CMOL cells. For example, for circuit s208, the number of tiles is 16 (i.e., 4 × 4); thus, the area available for use by circuit s208 is the area of 256 (i.e., 16 × 16) CMOL cells. In addition, the column "Cells" shows the number of CMOL cells used for NOR gates in the circuit netlist. The column "AU" indicates the area utilization; this is the number of cells occupied by a circuit, which is represented as a percentage of the number of CMOL cells used in the CMOL array territory. The column "Delay" shows the maximal logic level of the circuit netlist. The column "Buffers" represents the number of buffers inserted to satisfy the CMOL connectivity domain. The column "Time" shows the computation time required to perform cell allocation for a circuit in units of seconds.

In Table 2, the symbol "−" indicates that the approach cannot handle the circuit. The experimental results in Table 2 show that the tile-based approach (i.e., the CAD software tool [25]) cannot tackle cell allocation for hybrid CMOS/nanodevices with larger-scale circuits.

The experimental results in Table 3 demonstrate that the CGA and the method in this paper exhibit better area utilization than the CAD software tool. The required computational time can be reduced by about 34% by using the method in this paper compared to that for the CGA, and the number of buffers inserted is also decreased. In addition, for cell allocation in large logic circuits, the experimental results show that the delay can also be reduced by using the method in this paper.

The experimental results also show that the method in this paper can implement cell allocation for larger-scale circuits and can obtain an effective timing delay. An advantage of the method is that it can spend less time for cell allocation than the conventional CAD software tool and CGA.

6 Conclusions

Cell allocation is a basic and crucial part of the design and manufacture of CMOL circuits. The cell allocation performance is indicated by, for example, the area utilization and computation time. This paper presents a new method of cell allocation in CMOL circuits by using a cultural algorithm with chaotic behavior. Many experimental results show that the computation time required for this method is less than that of a CGA. Some work remains to be done in the future, such as selecting better parameter values for the evolution operations.

Acknowledgements This work was supported by Guangdong Provincial Natural Science Foundation of China (Grant No. 2014A030313441), Guangzhou Science and Technology Project (Grant No. 201510010169), Guangdong Province Science and Technology Project (Grant Nos. 2013B090600063, 2014B090901005, 2014A040401076), and the National Natural Science Foundation of China (Grant No. 61072028).

References

1. P. Lin, S. Pi, and Q. Xia, 3D integration of planar crossbar memristive devices with CMOS substrate, *Nanotechnology* 25(40), 405202 (2014)
2. N. P. Dasgupta and P. D. Yang, Semiconductor nanowires for photovoltaic and photoelectrochemical energy conversion, *Front. Phys.* 9(3), 289 (2014)
3. Z. L. Pan, L. Chen, G. Z. Zhang, and P. H. Wu, Detecting

the micro-defects in the GaAs materials by time resolved emissions, *Chin. Sci. Bull.* 59(16), 1838 (2014)

4. A. Sulaev, M. Zeng, S. Q. Shen, S. K. Cho, W. G. Zhu, Y. P. Feng, S. V. Eremeev, Y. Kawazoe, L. Shen, and L. Wang, Electrically tunable in-plane anisotropic magnetoresistance in topological insulator $BiSbTeSe_2$ nanodevices, *Nano Lett.* 15(3), 2061 (2015)
5. R. Tenne, Recent advances in the research of inorganic nanotubes and fullerene-like nanoparticles, *Front. Phys.* 9(3), 370 (2014)
6. Z. L. Pan, L. Chen, G. Z. Zhang, and P. H. Wu, A single-photon fault-detection method for nanocircuits that use GaN material, *Science China - Technological Sciences*, 57(2), 270 (2014)
7. C. Dong, W. Wang, and S. Haruehanroengra, Efficient logic architectures for CMOL nanoelectronic circuits, *Micro & Nano Lett.* 1(2), 74 (2006)
8. D. B. Strukov and K. K. Likharev, CMOL FPGA: A reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices, *Nanotechnology* 16(6), 888 (2005)
9. Z. Abid and M. Liu, 3D integration of CMOL structures for FPGA applications, *IEEE Trans. Comput.* 60(4), 463 (2011)
10. M. S. Zaveri and D. Hammerstrom, CMOL/CMOS implementations of bayesian polytree inference: Digital and mixed-signal architectures and performance/price, *IEEE Trans. NanoTechnol.* 9(2), 194 (2010)
11. A. Afifi, A. Ayatollahi, and F. Raissi, CMOL implementation of spiking neurons and spike-timing dependent plasticity, *International Journal of Circuit Theory and Applications* 39(4), 357 (2011)
12. Z. Abid, A. Almaaitah, M. Barua, and W. Wang, Efficient CMOL gate designs for cryptography applications, *IEEE Trans. NanoTechnol.* 8(3), 315 (2009)
13. G. Chen, X. Y. Song, and P. Hu, A theoretical investigation on CMOL FPGA cell assignment problem, *IEEE Trans. NanoTechnol.* 8(3), 322 (2009)
14. W. N. Hung, C. Gao, X. Song, and D. Hammerstrom, Defect-tolerant CMOL cell assignment via satisfiability, *IEEE Sens. J.* 8(6), 823 (2008)
15. Z. Chu, Y. Xia, W. N. Hung, and L. Wang, A memetic approach for nanoscale hybrid circuit cell mapping, *13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools*, Lille, France, pp 681–688 (2010)
16. Z. Chu, Y. Xia, L. Wang, and M. Hu, CMOL cell assignment based on dynamic interchange, *IEEE 8th International Conference on ASIC*, Changsha, China, 921–924 (2009)
17. X. J. Wang and L. Y. Wang, A pseudo-boolean programming approach for CMOL cell assignment, *IEEE International Conference on Electronics, Communications and Control*, Ningbo, China, pp 1265–1268 (2011)
18. M. H. Magnusson, B. J. Ohlsson, M. T. Bjork, K. A. Dick, M. T. Borgström, K. Deppert, and L. Samuelson, Semiconductor nanostructures enabled by aerosol technology, *Front. Phys.* 9(3), 398 (2014)
19. K. Shao, N. Ding, S. Huang, S. Ren, Y. Zhang, Y. Kuang, Y. Guo, H. Ma, S. An, Y. Li, and C. Jiang, Smart nanodevice combined tumor-specific vector with cellular microenvironment-triggered property for highly effective antiglioma therapy, *ACS Nano* 8(2), 1191 (2014)
20. H. Y. Fan, S. Wang, and L. Y. Hu, Evolution of the single-mode squeezed vacuum state in amplitude dissipative channel, *Front. Phys.* 9(1), 74 (2014)
21. M. Z. Ali and R. G. Reynolds, Cultural algorithms: A Tabu search approach for the optimization of engineering design problems, *Soft Comput.* 18(8), 1631 (2014)
22. Q. F. Luo, Y. Q. Zhou, P. G. Guo, and X. Chen, Functional network design using parallel cultural algorithm, *Appl. Math. Inf. Sci.* 8(4), 1949 (2014)
23. S. Srinivasan and S. Ramakrishnan, A social intelligent system for multi-objective optimization of classification rules using cultural algorithms, *Computing* 95(4), 327 (2013)
24. E. M. Sentovich, K. J. Singh, and L. Lavagno, SIS: A system for sequential circuit synthesis, Technical Report UCB/ERL M92/41, University of California, Berkeley, 1992
25. D. B. Strukov and K. K. Likharev, A reconfigurable architecture for hybrid CMOS/nanodevice circuits, *14th ACM International Symposium on Field Programmable*, New York, pp 131–140 (2006)