Satisfiability-based method for reconfiguring power efficient VLSI array

Junyan Qian\textsuperscript{a)}, Wei Cao, Jia Hu, Jingwei Zhang, Zhoubo Xu\textsuperscript{b)}, and Zhide Zhou

Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China

a) qjy2000@gmail.com
b) xzbli11@guet.edu.cn

Abstract: Fault-tolerant techniques are absolutely vital for large scale multiprocessor array as it suffers from frequent hardware defects or soft faults. This paper presents a satisfiability (SAT)-Based method for the reconfiguration of a two-dimensional degradable very-large-scale integration (VLSI) array with faulty processing elements (PEs). An SAT model of the target array is proposed such that the target array can be constructed by using the efficient SAT solver. For minimizing the interconnection length of the target array, we present an incomplete algorithm, to search a target array with suitable interconnection length to meet the system requirement. Our evaluations show that the proposed incomplete algorithm is efficient, which is compared with the state-of-the-art.

Keywords: reconfiguration, satisfiability, VLSI array, fault tolerance

Classification: Integrated circuits

References

[1] P. J. Chuang and L. C. Yao: “An efficient reconfiguration scheme for fault-tolerant meshes,” Inf. Sci. 172 (2005) 309 (DOI: 10.1016/j.ins.2004.05.003).
[2] I. Takanami and T. Horita: “A built-in circuit for self-repairing mesh-connected processor arrays by direct spare replacement,” IEEE 18th Pacific Rim International Symposium on Dependable Computing, PRDC 2012 (2012) 96 (DOI: 10.1109/PRDC.2012.11).
[3] S. Y. Kuo and I. Y. Chen: “Efficient reconfiguration algorithms for degradable VLSI/WSI arrays,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 11 (1992) 1289 (DOI: 10.1109/43.170991).
[4] C. P. Low and H. W. Leong: “On the reconfiguration of degradable VLSI/WSI arrays,” IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 16 (1997) 1213 (DOI: 10.1109/43.662684).
[5] C. P. Low: “An efficient reconfiguration algorithm for degradable VLSI/WSI arrays,” IEEE Trans. Comput. 49 (2000) 553 (DOI: 10.1109/12.862215).
[6] W. Jigang and T. Srikanthan: “An improved reconfiguration algorithm for degradable VLSI/WSI arrays,” J. Syst. Archit. 49 (2003) 23 (DOI: 10.1016/S1383-7621(03)00041-9).
[7] W. Jigang, et al.: “Preprocessing and partial rerouting techniques for accelerating reconfiguration of degradable VLSI arrays,” IEEE Trans. Very
1 Introduction

With the recent advances in VLSI technology, massively parallel systems with a huge number of processing elements (PEs) can be fabricated on a single chip or silicon wafer. Two-dimensional (2D) mesh-connected processor array (mesh array) is a type of massively parallel system and is known to be an efficient topology for many computational processes, such as matrix manipulation, image processing and signal processing. However, as the density of integration increases, the probability of defects occurring during the fabrication process also increases. In addition, it is almost impossible to guarantee all the PEs in the system to be fault-free throughout its working lifetime. Hence, efficient fault-tolerant technologies must be employed to enhance the system yield and reliability.

For reconfiguration of mesh array, there are generally two fault-tolerant methods, namely, the redundancy approach and the degradation approach. In the redundancy approach, the system is built with some spare PEs, which are used to directly replace the faulty PEs in the reconfigurable system. Varieties of techniques are described in the literatures to restructure a faulty physical system into a fault-free target logical system with spare rows and columns [1, 2]. However, the main drawback of this approach is the fixed size of a reconfigured array; that is, the system cannot be reconfigured and has to be discarded if the spare PEs cannot replace all the faulty PEs.
In the degradation approach, all the PEs of the systems are treated in the same way, and there are no spare PEs in the system. The fault-tolerance of the system is achieved by using as many fault-free PEs as possible to construct a logical mesh array. Kuo and Chen [3] studied the reconfiguration problem of 2D degradable mesh arrays consisting of four-port switches under three different routing constraints, called, 1) *row and column bypass*, 2) *row bypass column rerouting*, and 3) *row and column rerouting* to refactoring array. It was shown that most problems under different rerouting constraints are NP-complete. Low and Leong [4] proposed an optimal algorithm, named GCR, to construct a maximum target array (MTA) in linear time that contains the selected rows. Combined with the GCR algorithm, Low [5] proposed an efficient heuristic reconfiguration algorithm under the row and column rerouting constraint, which excludes the rows that form a bottleneck of the problem and using the fault-free PEs of these rows to compensate the faulty PEs of the neighbor rows. Jigang and Srikanthan [6] simplified the row-selection scheme for the rows to be excluded and proposed a partial rerouting scheme in [7] to accelerate the reconfiguration of the target array. Fukushi and Horiguchi, utilized genetic approach in [8] and [9] to construct the MTA.

As it is known, minimizing the total interconnection length (hereafter referred to as inter-length) of a target array leads to lesser routing costs, capacitance and dynamic power dissipation. To this end, a dynamic programming approach (denoted as DPGR in this paper) was introduced for reducing power dissipation of a logical array in [10] by reducing the number of long-interconnects. Furthermore, Jigang [11] proved that there exists at least one instance such that a MTA with minimum long-interconnects can be found in $\Omega(2^m)$ times at least. However, DPGR does not address minimizing the total interconnection length (hereafter referred to as inter-length) of the target array. Therefore, a divide-and-conquer algorithm (denoted as CMTA in this paper) was proposed for a tightly-coupled MTA in [12], resulting in significant improvements over DPGR in terms of the total inter-length. Qian et al. [13] proposed a method based on integer programming for constructing tightly-coupled subarrays. However, neither DPGR nor CMTA guarantees the resultant logical mesh array on the number of the long-interconnects is optimal. Still, no work addresses the exact model of the target array for constructing the power efficient target array.

Boolean Satisfiability (SAT) is a well-known decision problem. The goal of the satisfiability problem is to determine whether there exists an assignment of truth values to variables that makes the Conjunctive Normal Form (CNF) formula satisfiable. If no such assignment exists, the function expressed by the formula is FALSE for all possible variable assignments and the formula is unsatisfiable. SAT is a widely used modeling framework for solving combinatorial problems, such as combinational equivalence checking, automatic test-pattern generation, model checking, planning and haplotype inference [14].

In this work, we develop an SAT model of the target array to further reduce the inter-length. Then, using this model, we show that a target array can be constructed by employing efficient SAT solver. However, it is impossible to enumerate all available target array. Thus, for constructing the power efficient target array, we proposed an incomplete algorithm to search the target array such that the inter-
length of the target array meets the system requirement. Although the proposed incomplete algorithm does not search the whole solution space to find the optimal target array with the minimum number of long interconnects, it is able to have a good performance compared with the previous works.

In detail, we provide the following major contributions:

- We propose an SAT model of the target array.
- We show that the target array can be constructed by utilizing efficient SAT solver.
- We propose an incomplete algorithm for constructing the power efficient target array.

The rest of the paper is organized as follows: Section 2 presents the notation and definitions used in this paper. In section 3, we describe our SAT model and propose an incomplete algorithm for constructing the power efficient target array. In section 4, we present the experimental results and the performance comparison with previous works. Finally, we conclude our work in section 5.

2 Preliminaries

2.1 Architecture and reconfiguration schemes

The host array $H$ (degradable array) is defined as the original processor array after manufacturing, which may contain faulty PEs. Assume $N$ is the number of fault-free PEs in $H$, $\rho$ indicates the fault density of the host array, where $0 < \rho < 1$, i.e., there are $N = \rho \cdot m \cdot n$ faulty PEs and $O((1 - \rho) \cdot m \cdot n)$ valid interconnects in an $m \times n$ host array. The target array (logical array) $T$ is defined as the degradable subarray of $H$ that contains no faulty PEs. The rows (columns) in $H$ and $T$ are called physical rows (columns) and logical rows (columns), respectively. A target array $T$ sized $r \times s$ means $T$ contains $r$ logical rows and $s$ logical columns.

Fig. 1 shows the architecture of a $4 \times 4$ host array connected by four-port switches. Assuming $e_{i,j}$ stands for the PE in the $H$, where $i$ and $j$ are the row index and column index. In this paper, the assumptions of architecture are the same as that in [10], and [12]. Two basic reroute schemes are used to reconfiguration, called, row bypass scheme and column rerouting scheme.
As shown in Fig. 1, in row bypass scheme, $e_{i,j-1}$ can directly communicate with $e_{i,j+1}$ when $e_{i,j}$ is faulty. The data will bypass $e_{i,j}$ through an internal bypass link without being processed. In the column rerouting scheme, if $e_{i+1,j-1}$ is faulty, $e_{i,j}$ can connect directly to $e_{i+1,j'}$ with external switches, where $|j' - j| \leq d$, $d$ is called compensation distance [5]. As same as in [10] and [12], $d$ is also limited to 1.

As shown in Fig. 2, there are six possible types of link-ways for a target array [10], which can be classified into two classes based on the number of the switches used. One is called the short interconnect, which uses one switch to connect neighboring PEs; the other is called the long interconnect, which uses two switches. In Fig. 2, (a) and (d) are short interconnects, while the others are long interconnects. It is evident that minimizing the number of long interconnects in the target array without loss of harvest leads to lesser routing cost, capacitance, and dynamic power dissipation.

**Definition 1.** An MTA with the minimum number of the long interconnects is called the power efficient target array (PETA).

Fig. 3 shows two target arrays on a 4×4 host array with three faulty PEs. They are both MTA, but one is a common target array with five long interconnects, while the other is a PETA with only one long interconnect. In this paper, our goal is to find a solution for the following problem.
Problem $\mathcal{P}$ 1. Given an host array of size $m \times n$, find a PETA that contains the selected rows under the constraint of row and column rerouting scheme.

3 SAT method for Problem $\mathcal{P}$

In this section, we propose an efficient method to construct power efficient target array. Let $R_1, R_2, \cdots, R_m$ be the rows of the given host array. Without loss of generality, we assume that the target array contains the selected rows $R_1, R_2, \cdots, R_m$. Assume $B_l, B_r$ are two logical columns passing through each physical row of the $m \times n$ host array, where $B_l \leq B_r$ if the $i$th PE in $B_l$ lies to the left of, or is identical to, the $i$th PE in $B_r$, for $1 \leq i \leq m$. In this paper, $A[B_l, B_r]$ indicates the area that consists of the PEs bounded by $B_l$ and $B_r$ (including $B_l$ and $B_r$). The logical columns $B_l$ and $B_r$ are called the left and right boundaries of the area, respectively.

As is known that most satisfiability (SAT) solvers take the CNF (conjunctive normal form) file format as the input. Thus, we also use the conjunctive normal form to encode the proposed SAT model.

3.1 SAT model of target array

Suppose that the target array consisted of the logical columns $X_1, X_2, \cdots, X_k$ generated by GCR [4] in the left-to-right manner and the target array consisted of $Z_1, Z_2, \cdots, Z_k$ generated by GCR in the right-to-left manner (where $k$ is the total number of logical columns). As pointed out in the literature [10], these two boundaries $X_i$ and $Z_{k-i+1}$ are not independent, i.e., $X_i \leq Z_{k-i+1}$, and the area $A[X_i, Z_{k-i+1}]$ is the largest area available to generate the $i$th logical column.

Let $Y_i$ indicate the $i$th logical column located in the area $A[X_i, Z_{k-i+1}]$, where $Y_i$ consists of PE $P_1, P_2, \cdots, P_j$, $P_j \in R_j \cap A[X_i, Z_{k-i+1}]$, for $0 \leq j \leq m - 1$. Assume $E_j$ denotes the set of fault-free PEs in $R_j \cap A[X_i, Z_{k-i+1}]$, where the elements of $E_j$ are boolean variable as the fault-free PE in $R_j \cap A[X_i, Z_{k-i+1}]$ is used to form a logical column, or not. As mentioned in Section 2, the compensation distance is limited to 1, thus for any PE $u \in E_j$ and $v \in E_{j+1}$, if there exists an available interconnect between $u$ and $v$, $|\text{col}(u) - \text{col}(v)| \leq 1$ ($\text{col}(u)$ denotes the physical column index of the PE $u$). Meanwhile, exactly one element of $E_j$ is true because each logical column in the target array contains exactly one fault-free PE from each of the rows.

Assume $E_j = \{u_1, u_2, \cdots, u_\gamma\}$ and $E_{j+1} = \{v_1, v_2, \cdots, v_\eta\}$, where $\gamma$ and $\eta$ are the number of elements in the set $E_j$ and $E_{j+1}$, respectively. Intuitively, if $|\text{col}(u) - \text{col}(v)| \leq 1$ for any PE $u \in E_j$ and $v \in E_{j+1}$, we just need to guarantee that only one element is true in $E_j$ and $E_{j+1}$. However, there usually exist some PEs $u' \in E_j$ and $v' \in E_{j+1}$ such that $|\text{col}(u') - \text{col}(v')| > 1$. In this case, the interconnect between $u'$ and $v'$ need to be removed, i.e., $u'$ and $v'$ are not true at the same time.

Fig. 4 shows an example for encoding the logical column. The boolean variable in the box represents the fault-free PEs. The pairwise encoding method, which the most widely known encoding for the at-most-one constraint, is used to make sure that there is exactly one true element in $E_j$. For each pair of variables $(u, v)$, add a binary clause $\pi \lor \pi'$ that guarantees that only one of the two variables can be
assigned value true. However, it is possible that all variables are false in pairwise encoding method, which is inconsistent with the condition that the set $E_j$ must has a true variable. Thus, we add a clause that is a simple disjunctive formula of all variable in $E_j$, i.e., $\lor u_j$ for all $u_j \in E_j$, which is known as the at-least-one constraint. For example, $E_1 = R_1 \cap A[X_1, Z_2] = \{u_1, u_2, u_3\}$, the constraint that only one element in $E_1$ is true can be encoding as Eq. (1).

$$\lor \bar{u}_1 \lor \bar{u}_2 \lor \bar{u}_3 \lor \bar{u}_4 \lor \bar{u}_5 \lor \bar{u}_6 \lor \bar{u}_7 \lor \bar{u}_8$$

(1)

Hence, the completely description of the first logical column in Fig. 4(a) is as Eq. (2).

$$
\begin{array}{c|c|c}
\lor \bar{u}_1 & \lor \bar{u}_2 & \lor \bar{u}_3 \\
\lor \bar{u}_1 & \lor \bar{u}_5 & \lor \bar{u}_6 \\
\lor \bar{u}_2 & \lor \bar{u}_3 & \lor \bar{u}_7 \\
u_1 \lor u_2 \lor u_3 & u_4 & u_5 \lor u_6 \\
\end{array}
$$

(2)

As shown in Fig. 4(a), $u_5$ cannot connect to $u_9$ as $|col(u_5) - col(u_9)| > 1$, which makes $u_5$ and $u_9$ cannot be true at the same time, i.e., $\lor \bar{u}_5 \lor \bar{u}_9$. In Fig. 4(b), for each of the adjacent rows’ elements $v_i$ and $v_j$ ($1 \leq i, j \leq 7$) we have $|col(v_i) - col(v_j)| \leq 1$, so we should not add the constraint. The constraint can be encoding as Eq. (3).

$$\lor \bar{u}_5 \lor \bar{u}_9$$

(3)

However, the areas of each logical column always overlap each other, such that a fault-free PE is represented by many variables. For example, in Fig. 4, the PE $v_{0,2}$ has the possibility to belong to two logical column and it can be represented by $u_5$ and $v_1$. In this case, we need to guarantee that all variables represented the same fault-free PE only have at most one to be true as a fault-free PE can only belongs to a logical column. The constraint of overlapped elements can be encoding as Eq. (4).

$$\lor \bar{u}_5 \lor \bar{v}_7 \lor \bar{v}_8 \lor \bar{v}_1 \lor \bar{v}_6$$

(4)

What is more, the logical columns that we got cannot intersect. For example, if PE $u_6$ connect to $u_9$ in Fig. 4(a), PE $v_4$ cannot connect to $v_5$ in Fig. 4(b). In this case, we need to guarantee that all logical columns cannot intersect, the constraint can be encoding as Eq. (5).
Thus, the whole encoding of the example is described as Eq. (6), any truth assignment of the following formulas is a feasible solution of problem $\mathcal{P}$.

$$
\begin{align*}
\overline{u}_1 \vee \overline{u}_2 & \quad \overline{v}_1 \vee \overline{v}_2 \\
\overline{u}_1 \vee \overline{u}_3 & \quad \overline{v}_1 \vee \overline{v}_3 \\
\overline{u}_2 \vee \overline{u}_3 & \quad \overline{v}_2 \vee \overline{v}_3 \\
u_1 \vee u_2 \vee u_3 & \quad v_4 \\
u_4 & \quad \overline{v}_5 \vee \overline{v}_6 \\
\overline{u}_5 \vee \overline{u}_6 & \quad \overline{v}_5 \vee \overline{v}_7 \\
u_5 \vee u_6 & \quad \overline{v}_6 \vee \overline{v}_7 \\
\overline{u}_7 \vee \overline{u}_8 & \quad v_5 \vee v_6 \vee v_7 \\
\overline{u}_8 \vee \overline{u}_9 & \\
u_7 \vee u_8 \vee u_9
\end{align*}
$$

(6)

Hence, the formal overview of the constraints to produce a logical array is as follows:

1. The set $E_j$ exactly has one element to be true.
2. If there exists an interconnect between PE $u \in E_j$ and $v \in E_{j+1}$, $|\text{col}(u) - \text{col}(v)| \leq 1$.
3. If there does not exist an interconnect between PE $u' \in E_j$ and $v' \in E_{j+1}$, i.e., $|\text{col}(u') - \text{col}(v')| > 1$, then $u'$ and $v'$ are false simultaneously.
4. A fault-free PE can only be represented by one true variable at the same time.
5. The logical columns that we got should not intersect each other.

Notably, the second constraint is implicitly expressed by the (1) and (3) constraints. Assume that each $E_j$ has an average of $\theta$ elements. As it is known, the pairwise encoding method takes $O(\theta^2)$ clauses to encode the at-most-one constraint. Thus, we need

$$m \times k \times \theta^2 = m \times k \times \theta \times \theta \geq N \times \theta,$$

i.e., $O(N)$ clauses to encode the first condition. Suppose that $\phi$ interconnects among each $E_j$ are removed, thus $O(\phi)$ clauses are need to encode the third condition. Then suppose that $\delta$ logical columns have $\epsilon$ intersection, thus $O(\epsilon)$ clauses are need to encode the fifth condition. Finally, assume each fault-free is represented by $\eta$ variables on average, then we need $O(N \times \eta^2)$ clauses to encode the fourth condition. Above all, the SAT model needs $m \times k \times \theta$, i.e., $O(N)$ variables and $O(N \cdot (1 + \eta^2) + \phi + \epsilon)$ clauses to encode a target array.

### 3.2 Construct power efficient target array with SAT

In the previous section, we construct the SAT model of the target array. Any truth assignment of the SAT model is possible to be a PETA. However, it is impossible to obtain all solution of the SAT model as the number of solutions is growing exponentially with the increase of variables, which makes it hard to find the
global optimum solution for a target array with the minimum number of long interconnects (nlis). As reported in [11], there are $\Omega(2^m)$ candidates of a logical column in $A[X_i,Z_{k(i+1)}]$. Thus, we propose an incomplete method to find an MTA with the expected value $l$ of nlis, such that the target array meets the lower power consumption of the systems.

For an $m \times n$ host array and the expected value $l$ of nlis in the target array, $T_1$ and $T_2$ are the results generated by $GCR$ algorithm in left-to-right and right-to-left manners, respectively. The function $\text{CNF}(T_1, T_2)$ is used to produce the formulas $\varphi$ according to the modeling approaches described in previous section. Initially, we test whether $\varphi$ is satisfied, specially, there is at least one truth assignment of $\varphi$. If $\varphi$ is satisfied, we obtain a solution $S$. The process is repeated until $\varphi$ is unsatisfied or there is a solution $S$ such that the nlis of $S$ is less than $l$. If the nlis of a solution $S$ is great than $l$, we add, as a new clause, the negated form of $S$, i.e., $\varphi := \varphi \land \overline{S}$. When a solution $S$ that satisfy the requirements is obtained, we just need to map the variables of $S$ to the PEs for generating the finial power efficient target array. The formal description of this incomplete algorithm, denoted as $\text{CPETASAT}$, is as follows.

For example, a truth assignment of Fig. 4 is

$$\{u_2, u_4, u_6, v_2, v_3, v_4, v_6\},$$

which makes the target array with only one interconnect.

Algorithm 1: $\text{CPETASAT}(H, l, T)$;

| Input: an $m \times n$ host array $H$ and the expected value $l$ of the number of long interconnects. |
| Output: $T$-power efficient target array. |

$T_1 := \text{Gcr}^L(H)$; // $GCR$ in left-to-right.  
$T_2 := \text{Gcr}^R(H)$; // $GCR$ in right-to-left.  
$\varphi := \text{CNF}(T_1, T_2)$; // Generate CNF formulas.  
while $S := \text{SAT}(\varphi)$ do 
  if $nlis$ of $S < l$ then 
    break; 
  else 
    $\varphi := \varphi \land \overline{S}$; 
    $T := \text{map variables of } S \text{ to PEs}$; 

4 Experimental results

The proposed algorithm was implemented in C++ and the Cryptominisat-4.5.3 solver [15] was used as the back-end of the proposed method. All the options of Cryptominisat solver were set as default. For the performance comparison on $nlis$, we also implemented the algorithms $\text{CMTA}$ in C++. A Linux PC with an Intel(R) Core(TM)i5-4690 3.50 GHz CPU and 8.0 GB of memory was used to run the experiments. The corresponding performance is compared with each other on the same random input instances in our simulations, where the random faults are generated for host arrays with uniform distribution. In order to make a fair comparison, all the assumptions are the same as in [10] and [12].
In Table I, data collected for host arrays of different sizes from $16 \times 16$ to $20 \times 20$ and averaged over 10 random instances are shown. Four fault densities are used in the experiments: 10%, 15%, 20%, 25%. In the simulations, we set the value of nlis of CMTA to equal the expected value of CPETASAT as the results of CMTA are the best so far. The reconfiguration time of our algorithm is longer than CMTA, they are not comparable, which leads us not to show it in the table. Thus, the running time of CPETASAT is limited in 1800 seconds. The Iterations indicates the number of iterations of CPETASAT to find a target array with the expected number of long interconnects, which are equal to that of CMTA. In Table I, imp stand for the improvement in terms of nlis over CMTA. For example, imp of CMTA is calculated by

$$\left(1 - \frac{\text{nlis of CPETASAT}}{\text{nlis of CMTA}}\right) \times 100\%$$

### Table I. The results of the algorithm CMTA and CPETASAT, average of 10 random instances.

| Host array | Target array | Performance |
|------------|--------------|-------------|
| Size $m \times n$ | Fault $(\%)$ | Target array $m \times k$ | CMTA | CPETASAT | NumVars | NumClauses | Iterations | nlis | imp $(\%)$ |
| $16 \times 16$ | 10 | 16 x 11 | 33 | 0.047 | 566 | 2961 | 1034 | 32 | 3.03 |
| | 15 | 16 x 10 | 35 | 0.045 | 507 | 2937 | 1315 | 33 | 5.71 |
| | 20 | 16 x 9 | 34 | 0.042 | 451 | 2470 | 1502 | 32 | 5.82 |
| | 25 | 16 x 7 | 32 | 0.038 | 336 | 1526 | 1258 | 30 | 6.25 |
| $18 \times 18$ | 10 | 18 x 13 | 44 | 0.061 | 758 | 4542 | 829 | 42 | 4.55 |
| | 15 | 18 x 11 | 47 | 0.057 | 699 | 4132 | 851 | 45 | 4.26 |
| | 20 | 18 x 10 | 50 | 0.054 | 533 | 2545 | 962 | 49 | 2.00 |
| | 25 | 18 x 8 | 44 | 0.051 | 422 | 1884 | 900 | 43 | 2.28 |
| $20 \times 20$ | 10 | 20 x 14 | 52 | 0.074 | 997 | 6127 | 1057 | 51 | 1.92 |
| | 15 | 20 x 13 | 62 | 0.071 | 830 | 5401 | 1257 | 60 | 3.23 |
| | 20 | 20 x 11 | 56 | 0.064 | 718 | 4124 | 2351 | 54 | 3.57 |
| | 25 | 20 x 9 | 48 | 0.057 | 629 | 3907 | 2160 | 47 | 2.08 |

From Table I, it is clear that the nlis of CPETASAT are less than that of CMTA. For example, the nlis of CMTA is 32 for the $16 \times 16$ host array with 25% fault density, but the value of nlis is reduced to 30 for CPETASAT; thus, the improvement on nlis over CMTA is 6.25%, respectively. Obviously, the Iterations is increased with the increase of fault density except 25% fault density, because the whole number of the feasible target arrays is increased with the increase of fault density. When the fault density close to 25%, too many faulty processing elements result in the number of possible logical columns reduced, and eventually leads to Iterations decrease. Although the size of the target array is decreased with the increase of fault density, the area of a logical column is increased with the increase of fault density, which makes the number of the feasible logical columns to be also increased quickly. However, the target array consists of logical columns and the total number of feasible target array is the product of the number of each logical column. The quickly increased number of logical columns leads the solutions of feasible target array to be increased rapidly.
5 Conclusions

Power efficiency is one of the major considerations in VLSI systems. In this paper, for minimizing the interconnection length of the target array, we firstly have proposed a SAT model of the target array, which made the target array to be constructed by using the efficient SAT solver. Secondly, we have presented an incomplete algorithm, based on the proposed SAT model, to constructing the desired target array of a given interconnection length that meets the system requirement. Experimental results reveal that the proposed incomplete algorithm was capable of obtaining the better result compared with the state-of-the-art. In the future, we will consider to reduce the number of variables to minimize the size of the model. In addition, we will employ more strict constraints to reduce the size of the solution space of the target array for accelerating the reconfiguration of the power efficient target array.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under grant No. 61363030, 61562015 and 61572146, the High Level Innovation Team of Guangxi Colleges and Universities and Outstanding Scholars Fund, Guangxi Natural Science Foundation of China under grant No. 2014GXNSFAA118365, 2015GXNSFDA139038, Innovation Project of GUET Graduate Education (2016YJCX12), Guangxi Key Laboratory of Trusted Software Focus Fund, Program for Innovative Research Team of Guilin University of Electronic Technology.