SCIENCE CHINA
Information Sciences

• LETTER •

March 2019, Vol. 62 039108:1–039108:3
https://doi.org/10.1007/s11432-018-9474-3

High-efficient generation algorithm for large random active shield

Ruishan XIN[1,2], Yidong YUAN[1,2], Jiaji HE[1,2], Yuehui LI[1,2] & Yiqiang ZHAO[1,2*]

[1]*School of Microelectronics, Tianjin University, Tianjin* 300072*, China;*
[2]*Tianjin Key Laboratory of Imaging and Sensing Microelectronic Technology, Tianjin University, Tianjin* 300072*, China*

Received 10 January 2018/Accepted 29 May 2018/Published online 8 January 2019

Citation Xin R S, Yuan Y D, He J J, et al. High-efficient generation algorithm for large random active shield. Sci China Inf Sci, 2019, 62(3): 039108, https://doi.org/10.1007/s11432-018-9474-3

Dear editor,

Security integrated circuits play important roles in trusted platform modules [1], embedded devices, etc. However, invasive attacks [2] are serious threats on such circuits. Active shield is regarded as an effective method against invasive attacks. Active shield is a mesh of dense metal wires covering the surface of an integrated circuit (IC) [3]. The security level of IC increases with the geometric difficulty of the mesh. The active shield proposed in [4, 5] has a complex geometry based on random Hamiltonian path. Given this randomness, it is difficult to recognize the geometry of the shield. However, the available generation algorithms are agonizingly slow. In this letter, a high-efficient generation algorithm of random Hamiltonian path is proposed using a divide-and-conquer and dynamic programming optimized algorithm (DCDPOA). This algorithm has high execution efficiency and wide adaptability for large integrated circuits, while keeping good randomness.

Cycle merging algorithm. Cycle merging algorithm (CMA) is a generation algorithm described in [5]. The rationale of the CMA is the merging of small cycles to form a bigger cycle. In the beginning, several squares are formed. In each merging step, two cycles are randomly selected from the squares or already-merged cycles. If the selected cycles have parallel edges, they will be merged together. However, this selection is completely random and without control. Thus, as the area of the shield increases, the probability that the two selected cycles have parallel edges decreases quickly, which significantly reduces the efficiency. To improve the efficiency, the random selection procedure needs to be optimized.

High-efficient generation algorithm. Dynamic programming (DP) algorithm divides the original complicated problem into several related easier subproblems. By solving the subproblems recursively in order, DP algorithm finds a solution to the original problem [6]. Applying the rationale of DP algorithm to Hamiltonian path generation, the dynamic programming optimized algorithm (DPOA) is obtained. The steps of the DPOA are described in Algorithm 1 and also shown in Figure 1(a). Here, $S(i, j)$ stands for the square whose lower-left vertex coordinate is (i, j). CM(S) stands for the Hamiltonian path generated by the CMA, and S is a set of optional cycles. These cycles are merged into one path by the CMA.

The inputs are the width W and the height H of the needed Hamiltonian path P_F. Steps 1 and 2 generate a basic random Hamiltonian path by the CMA. In Step 1, sixteen squares form a cycle set S_B. In Step 2, based on S_B, a basic Hamiltonian path P_B is generated by the CMA. In Step 3, the parameters w and h, which stand for the width and height of P_B, respectively, are both initialized to eight. If W is equal or greater than H, Steps 5–14 are executed. In Step 6, a row of new squares, which form the extended cycle set S_{ER}, are generated over P_B. In Step 7, P_B and S_{ER} form a new bigger path by employing the

* Corresponding author (email: yq_zhao@tju.edu.cn)

CMA. This path then replaces the previous P_B. In Step 8, the height of P_B increases by 2, so h adds 2. Steps 5–9 will be executed repeatedly until h reaches H. Then, the upward extension finishes and the rightward extension begins. In Step 11, a column of new squares, which form the extended cycle set $S_{\rm EC}$, are generated on the right side of P_B. In Step 12, P_B and $S_{\rm EC}$ are merged by the CMA. In Step 13, the width w increases by 2. Steps 10–14 will be executed repeatedly until w reaches W. Then, the rightward extension finishes. At this point, P_B is the needed Hamiltonian path P_F. If W is less than H, the rightward extension is executed before the upward extension. Therefore, Steps 16–25 are executed.

Algorithm 1 DPOA

Input: width W and height H ($W, H \in 2\mathbb{N}$);
Output: final Hamiltonian path P_F;

1: $S_B \Leftarrow \{S(i,j) | 1 \leqslant i \leqslant 8, 1 \leqslant j \leqslant 8,\ i,j \in 2\mathbb{N}\}$;
2: $P_B \Leftarrow \mathrm{CM}(S_B)$;
3: $w \Leftarrow 8$, $h \Leftarrow 8$;
4: **if** $W \geqslant H$ **then**
5: **while** $h < H$ **do**
6: $S_{\rm ER} \Leftarrow \{S(i,h+1) | 1 \leqslant i \leqslant 8,\ i \in 2\mathbb{N}+1\}$;
7: $P_B \Leftarrow \mathrm{CM}(P_B, S_{\rm ER})$;
8: $h \Leftarrow h+2$;
9: **end while**
10: **while** $w < W$ **do**
11: $S_{\rm EC} \Leftarrow \{S(w+1,j) | 1 \leqslant j \leqslant H,\ j \in 2\mathbb{N}+1\}$;
12: $P_B \Leftarrow \mathrm{CM}(P_B, S_{\rm EC})$;
13: $w \Leftarrow w+2$;
14: **end while**
15: **else**
16: **while** $w < W$ **do**
17: $S_{\rm EC} \Leftarrow \{S(w+1,j) | 1 \leqslant j \leqslant 8,\ j \in 2\mathbb{N}+1\}$;
18: $P_B \Leftarrow \mathrm{CM}(P_B, S_{\rm EC})$;
19: $w \Leftarrow w+2$;
20: **end while**
21: **while** $h < H$ **do**
22: $S_{\rm ER} \Leftarrow \{S(i,h+1) | 1 \leqslant i \leqslant W,\ i \in 2\mathbb{N}+1\}$;
23: $P_B \Leftarrow \mathrm{CM}(P_B, S_{\rm ER})$;
24: $h \Leftarrow h+2$;
25: **end while**
26: **end if**
27: $P_F \Leftarrow P_B$.

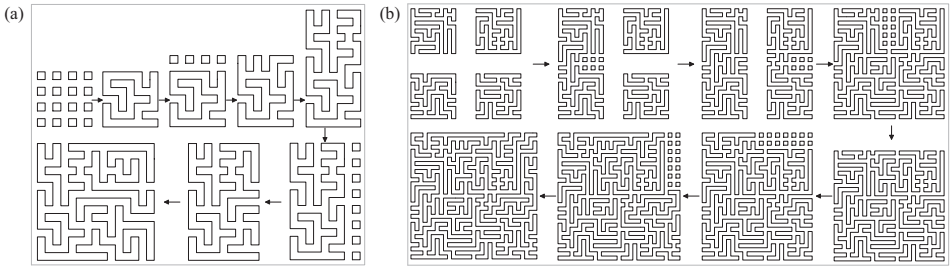
Divide-and-conquer (DC) algorithm is also applied to full-chip shield generation. DC algorithm decomposes a problem into disjoint subproblems, solves each subproblem recursively, and eventually composes their solutions to solve the original problem [6]. In this case, large path generation is divided into several subpath generations which are realized by the DPOA. Thus, the DCDPOA combines DC and DP algorithms. The steps are shown in Algorithm 2 and in Figure 1(b). $\mathrm{DP}(w,h)$ is defined as the path generated by the DPOA.

Algorithm 2 DCDPOA

Input: width W and height H ($W, H \in 2\mathbb{N}$);
Output: final Hamiltonian path P_F;

1: Solve $\begin{cases} w \times N + 4 \times (N-1) + p = W, \\ N \in \mathbb{N},\ w, p \in 2\mathbb{N}, \end{cases}$
and $\begin{cases} h \times M + 4 \times (M-1) + q = H, \\ M \in \mathbb{N},\ h, q \in 2\mathbb{N}, \end{cases}$
get suitable N, w, p, M, h, q;
2: **for** $n = 1 : N$ **do**
3: **for** $m = 1 : M$ **do**
4: $P_a \Leftarrow \mathrm{DP}(w,h)$;
5: **if** $m == 1$ **then**
6: $P_b \Leftarrow P_a$;
7: **else**
8: $S_{\rm CH} \Leftarrow \{S(i,j) | 1 \leqslant i \leqslant w, h \leqslant j \leqslant h+4, i,j \in 2\mathbb{N}+1\}$;
9: $P_b \Leftarrow \mathrm{DP}(P_a, S_{\rm CH}, P_b)$;
10: **end if**
11: **end for**
12: **if** $n == 1$ **then**
13: $P_c \Leftarrow P_b$;
14: **else**
15: $S_{\rm CV} \Leftarrow \{S(i,j) | 1 \leqslant j \leqslant [h \times M + 4 \times (M-1)], w \leqslant i \leqslant w+4,\ i,j \in 2\mathbb{N}+1\}$;
16: $P_c \Leftarrow \mathrm{DP}(P_b, S_{\rm CV}, P_c)$;
17: **end if**
18: **end for**
19: $S_{\rm RV} \Leftarrow \{S(i,j) | 1 \leqslant i \leqslant [w \times N + 4 \times (N-1)], [h \times M + 4 \times (M-1)] \leqslant j \leqslant H, i,j \in 2\mathbb{N}+1\}$;
20: $P_c \Leftarrow \mathrm{DP}(P_c, S_{\rm RV})$;
21: $S_{\rm RH} \Leftarrow \{S(i,j) | [w \times N + 4 \times (N-1)] \leqslant i \leqslant W, 1 \leqslant j \leqslant H,\ i,j \in 2\mathbb{N}+1\}$;
22: $P_c \Leftarrow \mathrm{DP}(P_c, S_{\rm RH})$;
23: $P_F \Leftarrow P_c$.

In Step 1, the target path is divided into an array of subpaths with the size of M rows and N columns. The width and height of each subpath are w and h, respectively. Two subpaths need 4 columns of vertices to combine in the horizontal direction or 4 rows in the vertical direction. The remaining vertices shown as p and q, which cannot form subpaths, are merged separately in the last few steps. The equation sets are solved to obtain the suitable values of N, w, p, M, h, and q. The parameters p and q need to be as small as possible. From Steps 2–18, the subpaths are generated and combined. M subpaths are combined in the vertical direction to form a high subpath. Then, N high subpaths are combined in the horizontal direction to form a wide path. High subpath generation is realized from Steps 3–11. In such a generation, M subpaths are needed. In Step 4, a subpath P_a is generated by the DPOA. If P_a is generated for the first time, the combination of subpaths, P_b, is initialized to P_a in Step 6. Otherwise, P_a needs to be combined with P_b. In Step 8, $S_{\rm CH}$ is generated to connect P_a and P_b. In Step 9, P_a, P_b, and $S_{\rm CH}$ are combined by employing the DPOA. After M cycles, a high subpath P_b is generated. Likewise, N high subpaths are combined in the horizontal



(c)

Size	CMA		DPOA		DCDPOA		SUR
	ET (s)	$E(d)$ (bit)	ET (s)	$E(d)$ (bit)	ET (s)	$E(d)$ (bit)	
400	12.2	0.965	1.61	0.946	1.63	0.999	7.48
900	129.4	0.991	2.75	0.993	3.03	0.992	42.71
1600	697.9	0.988	3.6	0.986	3.28	0.994	212.77
2500	2454.4	0.997	5.84	0.994	3.5	0.996	701.26
3600	7620.3	0.995	11.52	0.994	6.31	0.995	1207.65
1.0E+04	N/A	N/A	71.02	0.99	10.84	0.996	N/A
4.0E+04	N/A	N/A	1126.97	0.989	40.37	0.995	N/A
9.0E+04	N/A	N/A	5907.6	0.989	79.55	0.995	N/A
1.0E+06	N/A	N/A	N/A	N/A	1050.07	0.996	N/A
4.0E+06	N/A	N/A	N/A	N/A	8110.54	0.995	N/A

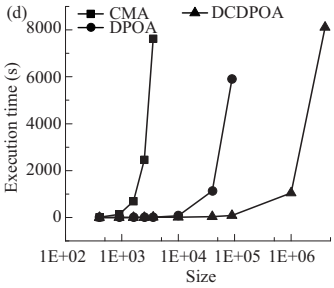


Figure 1 (a) Procedure of DPOA; (b) procedure of DCDPOA; (c) simulation results; (d) ET of three algorithms.

direction to obtain a wide path P_c from Steps 12–17. In Step 19, q rows of the remaining vertices form S_{RV}. In Step 20, S_{RV} is combined with P_c. In Step 21, p columns of the remaining vertices form S_{RH}. S_{RH} is merged with P_c in Step 22. The final P_c is the needed path P_F. The generation process finishes.

Simulation results. The DCDPOA, DPOA, and CMA are all implemented on the MATLAB2016a platform on Linux on a computer with an Intel Xeon X5690 CPU cadenced at 3.47 GHz. We introduce the entropy as the evaluating indicator of shield quality [4]. For a 2-dimensional shield, it is estimated as

$$E(d) = \sum_{d\in\{x,y\}} -P(d)\cdot\log_2 P(d), \qquad (1)$$

where $P(d)$ is the probability for the path to take this direction. The optimal value is 1.000 bit for a 2-dimensional shield.

Simulation results are given in Figure 1(c), where size is the number of vertices. The execution time (ET) acquired from MATLAB2016a is utilized as the main indicator. The speedup ratio (SUR) is the ratio of the ET of the CMA to the ET of the DCDPOA in the path generation process of the same size. The Hamiltonian paths generated by the three algorithms have similar entropy values. According to the SUR, the DCDPOA is more efficient than the CMA. Meanwhile, the DCDPOA is capable of large shield generation. For a 0.18 μm CMOS process with a thick top metal layer, the minimal width and space of the top metal are both 1.5 μm. Thus, to protect a chip with an area of 3 mm×3 mm, a shield with 1×10^6 vertices is needed. Owing to the DCDPOA, this shield can be generated within 20 min. The ET of the three algorithms is shown in Figure 1(d). All the curves show exponential growth. The CMA curve shows phenomenal growth when the size is over 1600, while a similar growth appears when the size is up to 1×10^6 on the curve of the DCDPOA.

Conclusion. This study proposes a high-efficient generation algorithm for random active shield. By employing DP and DC algorithms, the execution speed of the DCDPOA is seven times faster than that of the CMA. The DCDPOA is capable of full-chip shield generation, thus improving the security of the whole chip.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 61376032) and Tianjin Science and Technology Project of China (Grant No. 15ZCZDGX00180).

References

1 Shen C X, Zhang H G, Feng D G, et al. Survey of information security. Sci China Ser F-Inf Sci, 2007, 50: 273–298
2 Tria A, Choukri H. Invasive attacks. In: Encyclopedia of Cryptography & Security. Boston: Springer, 2011. 623–629
3 Xuan T N, Danger J L, Guilley S, et al. Cryptographically secure shield for security IPs protection. IEEE Trans Comput, 2017, 66: 354–360
4 Briais S, Cioranesco J M, Danger J L, et al. Random active shield. In: Proceedings of Workshop on Fault Diagnosis and Tolerance in Cryptography, Piscataway, 2012. 103–113
5 Briais S, Caron S, Cioranesco J M, et al. 3D hardware canaries. In: Cryptographic Hardware and Embedded Systems. Berlin: Springer, 2012. 1–22
6 Cormen T H, Leiserson C E, Rivest R L, et al. Introduction to Algorithms. 3rd ed. Cambridge: MIT Press, 2009. 65–97, 359–397