Fault Diagnosis of Arbiter Physical Unclonable Function

Jing Ye[1], Qingli Guo[1,2], Yu Hu[1], Xiaowei Li[1]
[1] State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences
[2] University of Chinese Academy of Sciences

ABSTRACT

Physical Unclonable Function (PUF) has broad application prospects in the field of hardware security. If faults happen in PUF during manufacturing, the security of whole chip will be threatened. Fault diagnosis plays an important role in the yield learning process. However, since different manufactured PUFs with the same design have different Challenge-Response Pairs (CRPs), which cannot be predicted, the traditional fault diagnosis method based on comparing the fault-free responses of a design and the failing responses of chips is no longer suitable for diagnosing PUF. Therefore, this paper proposes a fault diagnosis method toward classic arbiter PUF. The stuck-at faults and the delay faults are considered. Based on the expected uniformity of arbiter PUF, a diagnostic challenge generation method and a corresponding CRP analysis method are proposed to distinguish faults within the arbiter PUF. Experimental results show that the diagnostic accuracy achieves 100.0% with good diagnostic resolution.

Keywords

Arbiter Physical Unclonable Function; Stuck-At Fault; Delay Fault; Fault Diagnosis; Diagnostic Challenge Generation

1. Introduction

Physical Unclonable Function (PUF) is an emerging hardware security primitive [1]. It exploits the random physical disorder or the process variations to output particular responses for input challenges, which are called the *Challenge-Response Pairs (CRPs)*. PUF has broad application prospects in the field of hardware security, such as authentication, IP protection, and hardware obfuscation.

In general, PUFs can be broadly classified into two categories: the weak PUF and the strong PUF. The weak PUF normally has only a small number of CRPs, such as coating PUF [2], SRAM PUF [3], Memristor PUF [4], and DRAM PUF [5].

The strong PUF normally has numerous CRPs. A classic strong PUF is the arbiter PUF [6], whose responses depend on the comparison of the delays of two paths. Each path can dynamically consist of many path segments, and which path segments are included in a path at one time are determined by the challenge. Beyond it, various arbiter PUFs such as the XOR arbiter PUF [7], the lightweight arbiter PUF [8], and the obfuscation arbiter PUF [9] are developed lately.

This paper is supported by the *National Natural Science Foundation* (*NSFC*) of China: 61532017, 61274030, 61376043.

During manufacturing, faults may happen in the PUFs. If the PUFs do not function as expected, the security of whole chip will be threatened. In weak PUFs, such as memory cell based PUFs, every memory cell produces one response bit, so a single fault normally affects only one response bit. But in strong PUFs, such as the arbiter PUF, a single fault can affect a large amount of CRPs.

Like general digital IC, where stuck-at fault may occur in combinational logic, stuck-at fault may also occur within arbiter PUF. Moreover, arbiter PUF is more sensitive to delay fault. In combinational logic, if delay fault is not at the critical paths, the chip can still function well, and even if small delay fault is at the critical paths, the chip may still be used through degradation for guaranteeing the yield. However, in arbiter PUF, delay fault can result in serious loss of PUF security property, and even degradation cannot help to repair failing arbiter PUF, so it is an important source of yield loss.

Some previous works have studied how to test the classic arbiter PUFs [10][11]. But from the aspect of yield learning, exploring the root causes of faults to guide the improvement of design, placement, and routing of PUFs or manufacturing process is also important. Fault diagnosis [12]-[18] plays an important role in the yield learning process. It reports the most possible faults in a failing chip to help the physical failure analysis and the yield learning.

Fault diagnosis methods toward digital ICs have been studied for years. However, traditional fault diagnosis methods toward digital ICs are not suitable for diagnosing PUFs. This is because traditional fault diagnosis methods are based on comparing the fault-free responses of a design with the failing responses of chips. But for PUFs, their CRPs depend on process variations. Different manufactured PUFs with the same design may have different CRPs. In other words, even designers do not know the exact fault-free CRPs of manufactured PUFs. Therefore, the traditional fault diagnosis methods are no longer available.

To handle this issue, we propose a fault diagnosis method toward the classic arbiter PUF. The major contributions of this paper include:

(1) This is the first paper that diagnoses stuck-at faults and delay faults of arbiter PUF;

(2) A diagnostic challenge generation method and a corresponding CRP analysis method are proposed to distinguish the faults of arbiter PUF;

(3) Experiments on a large amount of fault diagnosis instances show the diagnostic accuracy achieves 100.0%, while the average diagnostic resolution achieves 1.7.

The rest of paper is organized as follows. Section 2 reviews the arbiter PUF. Section 3 introduces the target faults and the proposed fault diagnosis method. The experimental results are given in Section 4. Final is the conclusion.

2. Arbiter PUF

The circuit of arbiter PUF with 32-bit challenge c_1~c_{32} and 1-bit response r is shown in Fig.1. This PUF has totally $2^{32}=4.3\times10^9$ CRPs. When a challenge is prepared, a transition will be transmitted from t to r through two paths. If $c_k=0$ ($k\in[1,32]$), the transition goes from a_{k-1} to a_k through p_k, and goes from b_{k-1} to b_k through s_k; If $c_k=1$, the transition goes from a_{k-1} to b_k through q_k, and goes from b_{k-1} to a_k through r_k. Finally, the arbiter compares which transition arrives at arbiter earlier to produce the response.

This paper adopts the D flip-flop as the arbiter, and assumes it has both *SET* and *RESET* ports which are required in our diagnosis method to distinguish faults. In the following context, we will use the arbiter PUF with 32-bit challenge in Fig.1 as illustration. The proposed fault diagnosis method is suitable for arbiter PUFs with other number of challenge bits.

Please notice that, in application, responses of arbiter PUF may not be directly accessible for security [7-9]. However, during test and diagnosis, fuse can be used for accessing the responses, and after that, fuse can be burned so that no one can directly access the responses anymore [19].

3. Fault Diagnosis

3.1 Fault

This paper targets the stuck-at faults and the delay faults. For a net f with stuck-at v fault, we use f/v to represent it, which indicates the logic value of f is always v. For a net f with delay fault, we use f/T to represent it, which indicates the actual delay of f is much larger than its nominal delay. We assume only one fault happens in one failing arbiter PUF. All the suspect faults of arbiter PUF in Fig.1 are shown in Fig.2, classified into five sets: S_1~S_5.

S_1~S_4 contain stuck-at faults that happen at every net of arbiter PUF, except $i_1/0$, $i_1/1$, $j_1/0$, and $j_1/1$. Fig.3 illustrates the fault behaviors of $i_1/0$. Because i_1 is close to the t, where the transition starts, c_1 can work similarly as the fault-free situation even when $i_1/0$ happens. Therefore, we do not consider them as the suspect faults, and do not include them in the suspect fault list.

S_5 contains the delay faults, while the delay faults that happen at t, i_k, j_k, and c_k ($k\in[1,32]$) are not included in the suspect fault list. Obviously, the delay fault at t will not bring much effect to the arbiter PUF. The delay faults at i_k, j_k, and c_k can be repaired by slowing down the working frequency. Thus, they are not included in suspect faults.

3.2 Diagnosis

The target of fault diagnosis is that, given a failing arbiter PUF, finding out which of the suspect faults are most likely to be the actual fault. The faults reported by diagnosis are called the candidate faults. The candidate faulty nets are expected as accurate as possible and as few as possible.

The fault diagnosis flow is shown in Fig.4. Diagnostic challenges are generated firstly. For one PUF design, diagnostic challenges are only generated once. Then the failing responses of the to-be-diagnosed failing PUF are collected. Finally, by analyzing the failing CRPs, candidate faults are obtained.

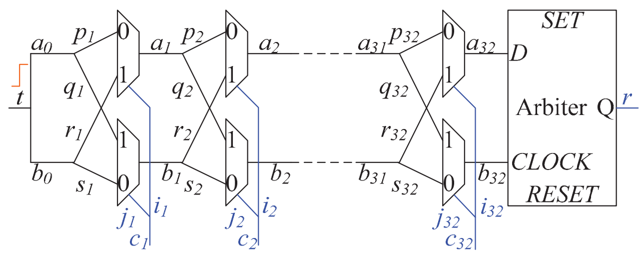

Fig.1 Arbiter PUF

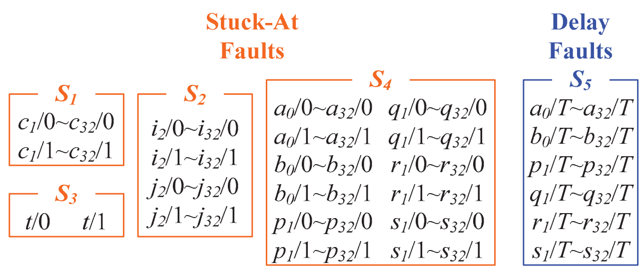

Fig.2 Fault

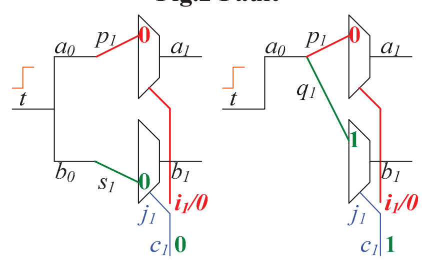

Fig.3 Fault Behaviors of $i_1/0$

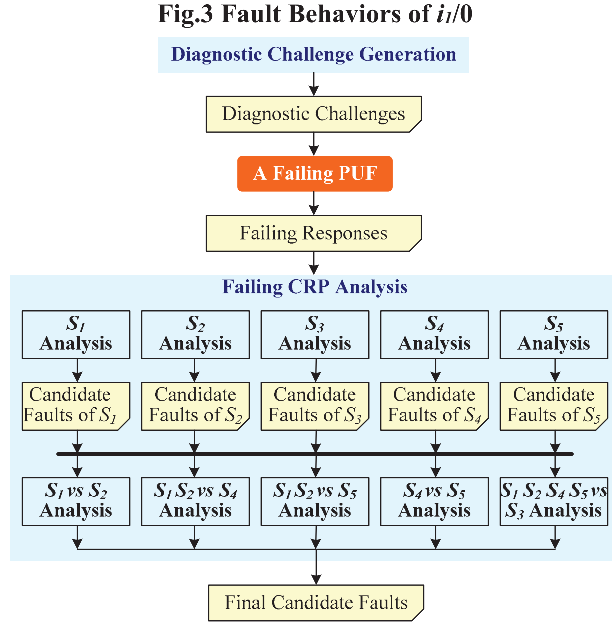

Fig.4 Fault Diagnosis Flow

As explained before, the fault-free CRPs of PUFs are unknown. Hence, the traditional fault diagnosis method is not suitable for PUF. To handle this issue, our method is based on two ideas:

(1) Though the fault-free CRPs are unknown, the expected uniformity of CRPs is known. The uniformity evaluates the probability of responses to be 0 or 1. The ideal uniformity of a fault-free PUF is 50%. This is leveraged in the proposed fault diagnosis method.

(2) Every diagnostic challenge is input to the failing PUF twice. For one time, the arbiter flip-flop is initialized to 0 by *RESET* port, and then the challenge is input to produce the response. For the other time, the arbiter flip-flop is initialized to 1 by *SET* port. In other words, every diagnostic challenge will obtain two responses. We use Q_0 to represent the initialized value.

In the following subsections, as shown in Fig.4, we will first propose how to generate diagnostic challenges and analyze failing CRPs to distinguish the suspect faults belong to the same set of $S_1 \sim S_5$. After obtaining the candidate faults of each set, we will propose how to distinguish the faults belong to different sets to obtain the final candidate faults.

3.2.1 S_1

The fault behaviors of S_1 are shown in Fig.5. If c_k occurs stuck-at 0 fault, the transition will always go from a_{k-1} to a_k through p_k, and go from b_{k-1} to b_k through s_k, no matter what value c_k is. In other words, if the stuck-at fault occurs at c_k (c_k/0 or c_k/1), the two challenges $\{c_1=v_1, c_2=v_2, ..., c_{k-1}=v_{k-1}, c_k=0, c_{k+1}=v_{k+1}, ..., c_{32}=v_{32}\}$ and $\{c_1=v_1, c_2=v_2, ..., c_{k-1}=v_{k-1}, c_k=1, c_{k+1}=v_{k+1}, ..., c_{32}=v_{32}\}$ ($v_k \in [0,1]$) will always produce the same response. However, if these two challenges produce the same response for a failing PUF, it does not mean there must be a stuck-at fault at c_k.

Hence, we adopt a probabilistic calculation method. For each suspect faulty net c_k of S_1, we generate n_1 such diagnostic challenge pairs. If the number of challenge pairs, whose two challenges produce the same response, is m_1, then the probability that c_k is the actual faulty net is calculated as m_1/n_1.

According to the uniformity of arbiter PUF, if n_1 is large enough, only the actual faulty net can obtain 100% probability. In such way, different suspect faulty nets of S_1 can be distinguished from each other. The suspect faulty nets with 100% probability are selected as the candidate faulty nets of S_1.

3.2.2 S_2

The fault behaviors of S_2 are shown in Fig.6. If i_k occurs the stuck-at 0 fault, the transition will always go from a_{k-1} to a_k through p_k. In such case, if c_k=0, the other transition will go from b_{k-1} to b_k through s_k as normal. But if c_k=1, one transition will go from a_{k-1} to both a_k and b_k, while the other transition is bypassed. In other words, the values of $c_1 \sim c_{k-1}$ will no longer affect the response. Notice that, j_k/0 also has this fault behavior, so i_k/v is indistinguishable from j_k/v, which is an inherent limitation of arbiter PUF.

Based on this characteristic, for each suspect fault i_k/v or j_k/v of S_2, we generate n_2 diagnostic challenge pairs. In each challenge pair, the value of c_k of two challenges is 1-v, the values of $c_{k+1} \sim c_{32}$ of one challenge is the same as those of the other challenge, and at least one value of $c_1 \sim c_{k-1}$ of one challenge is different from that of the other challenge. If the number of challenge pairs, whose two challenges produce the same response, is m_2, then the probability that i_k/v or j_k/v is the actual fault is calculated as $=m_2/n_2$.

Assume the actual fault is i_a/v (j_a/v). According to the uniformity of arbiter PUF, if n_2 is large enough, only i_a/v and j_a/v can obtain 100% probability. In such way, suspect faults of S_2 with different k can be distinguished from each other. The suspect faults with 100% probability are selected as the candidate faults of S_s.

3.2.3 S_3

The fault behaviors of S_3 are shown in Fig.7. If t occurs the stuck-at fault, no transitions can be propagated to either D or *CLOCK*. Hence, Q will be always the same as Q_0. Therefore, only if all the generated diagnostic challenges satisfy this characteristic, t is selected as the candidate faulty net of S_3.

3.2.4 S_4

The fault behaviors of S_4 are shown in Fig.8. In arbiter PUF, the transition is propagated through two paths to arrive at D and *CLOCK* of the arbiter flip-flop, respectively. These two paths contain different path segments a_k, b_k, p_k, q_k, r_k, and s_k. There are three scenarios:

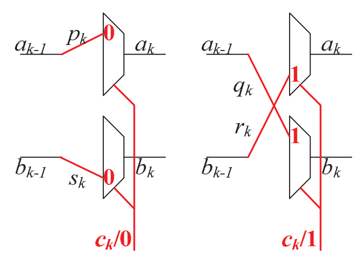

Fig.5 Fault Behaviors of S_1

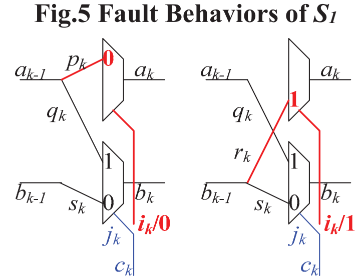

Fig.6 Fault Behaviors of S_2

Row	D	CLOCK	Q_0	Q
1	0	0	0	0
2	0	0	1	1
3	1	1	0	0
4	1	1	1	1

Fig.7 Fault Behaviors of S_3

Row	D	CLOCK	Q_0	Q
1	0	↑	0	0
2	0	↑	1	0
3	1	↑	0	1
4	1	↑	1	1
5	↑	0	0	0
6	↑	0	1	1
7	↑	1	0	0
8	↑	1	1	1
9	↑	↑	0	0 or 1
10	↑	↑	1	0 or 1

Fig.8 Fault Behaviors of S_4

(1) The stuck-at fault is at the path to D, so the transition only arrives at $CLOCK$, as shown in rows 1~4;

(2) The stuck-at fault is at the path to $CLOCK$, so the transition only arrives at D, as shown in rows 5~8;

(3) The stuck-at fault is at neither of the two paths, as shown in rows 9 and 10.

This truth table helps us to speculate which path the actual fault is at. For example, we set Q_0 to 0, and then we input a challenge and assume the Q keeps 0. Then, we set Q_0 to 1 and input the same challenge. If the Q remains 1, there is only one possibility: there is a stuck-at fault happens at the path to $CLOCK$. If the Q is changed to 0, there are two possibilities: 1) the response of this challenge is indeed 0, and 2) there is a stuck-at 0 fault happens at the path to D.

It is impossible to directly recognize the actual possibility, so we still adopt the probabilistic calculation method. For S_4, certain diagnostic challenges are generated. Among the challenges, for each suspect fault f/v, n_{4c} challenges make f at the path to $CLOCK$, and n_{4d} challenges make f at the path to D. If among n_{4c} challenges, m_{4c} challenges satisfy "when Q_0=0, Q=0" and "when Q_0=1, Q=1", and among n_{4d} challenges, m_{4d} challenges satisfy: "when Q_0=v, Q=v" and "when Q_0=1-v, Q=v", the probability that f/v is the actual fault is calculated as $(m_{4d}+m_{4c})/(n_{4d}+n_{4c})$.

Based on this calculation formula, to distinguish two suspect faulty nets f_1 and f_2 of S_4, there must be a challenge making that: f_1 arrives at D ($CLOCK$) while f_2 does not arrive at D ($CLOCK$). Then according to the uniformity of arbiter PUF, if number of such challenges is large enough, only the actual faulty net can obtain 100% probability.

However, not every suspect faulty net can satisfy it. Due to the circuit structure of arbiter PUF, if a challenge makes p_k, r_k, p_{k+1}, or q_{k+1} arrives at D ($CLOCK$), a_k always arrives at D ($CLOCK$) too; if a challenge makes q_k, s_k, r_{k+1}, and s_{k+1} arrives at D ($CLOCK$), b_k always arrives at D ($CLOCK$) too. To handle this issue, we observe that, if the actual faulty net is a_k, the probability of a_k, p_k, r_k, p_{k+1}, and q_{k+1} will all be 100%. But if the actual faulty net is p_k, r_k, p_{k+1}, or q_{k+1}, the probability of a_k may not be 100% and only the probability of p_k, r_k, p_{k+1}, or q_{k+1} is 100%. Therefore, during diagnosis, if all of a_k, p_k, r_k, p_{k+1}, and q_{k+1} obtain 100% probability, only a_k is selected as the candidate faulty net of S_4. Similarly, if all of b_k, q_k, s_k, r_{k+1}, and s_{k+1} obtain 100% probability, only b_k is selected as the candidate faulty net of S_4. Otherwise, the suspect faults with 100% probability are selected as the candidate faults of S_4.

3.2.5 S_5

The fault behaviors of S_5 is that: if the delay fault is at the path to D, Q will always be 0 no matter Q_0 is 0 or 1; if the delay fault is at the path to $CLOCK$, Q will always be 1 no matter Q_0 is 0 or 1.

For S_5, certain diagnostic challenges are generated. Among the challenges, for each suspect fault f/T, n_{5c} challenges make f at the path to $CLOCK$, and n_{5d} challenges make f at the path to D. If among n_{5c} challenges, m_{5c} challenges produce Q=1, and among n_{5d} challenges, m_{5d} challenges produce Q=0, the probability that f/T is the actual fault is calculated as $(m_{5d}+m_{5c})/(n_{5d}+n_{5c})$.

This calculation formula indicates that, to distinguish two suspect faults f_1/T and f_2/T of S_5, there must be a challenge making that: f_1 arrives at D ($CLOCK$) while f_2 does not arrive at D ($CLOCK$). Then according to the uniformity of arbiter PUF, if the number of such challenges is large enough, only the actual fault can obtain 100% probability.

Similarly as S_4, if all of a_k/T, p_k/T, r_k/T, p_{k+1}/T, and q_{k+1}/T obtain 100% probability, only a_k/T is selected as the candidate fault of S_5; if all of b_k/T, q_k/T, s_k/T, r_{k+1}/T, and s_{k+1} $/T$ obtain 100% probability, only b_k/T is selected as the candidate fault of S_5. Otherwise, the suspect faults with 100% probability are selected as the candidate faults of S_5.

3.2.6 S_1 vs S_2

Most suspect faults of S_1 can be easily distinguished from the suspect faults of S_2, since they have different fault behaviors. If the actual fault belongs to S_1, the suspect faults in S_2 will not be likely to obtain the 100% probability. If the actual fault belongs to S_2, the suspect faults in S_1 will not be likely to obtain the 100% probability as well. If some suspect faults of S_1 and S_2 indeed obtain 100% probability simultaneously, they are all kept in the final candidate faults except the following situation.

The considered exception is the suspect faulty nets c_1 of S_1, and i_2 and j_2 of S_2. The diagnostic challenge pairs for them all require the two challenges of each pair to have different values of c_1, but the diagnostic challenge pairs for i_2/v and j_2/v further require the two challenges of each pair to have the same value of c_2: 1-v. In such case, if the actual faulty net is c_1, the probability of c_1, i_2, and j_2 will all be 100%. But if the actual faulty net is i_2 or j_2, the probability of c_1 may not be 100%. Therefore, if c_1, i_2 and j_2 are selected as candidate faulty nets simultaneously, i_2 and j_2 are deleted and only c_1 is kept in the final candidate faulty nets.

3.2.7 S_1, S_2 vs S_4

It is possible that the suspect faults of S_1 or S_2 and the suspect faults of S_4 obtain 100% probability simultaneously, so the probability are insufficient to distinguish the suspect faults of S_1 and S_2 from the suspect faults of S_4. To effectively distinguish them, we use the characteristic that the suspect faults of S_4 can lead to the situation: "when Q_0=0, Q=0" and "when Q_0=1, Q=1", while the suspect faults of S_1 and S_2 cannot. If this situation happens, the candidate faults belong to S_1 and S_2 are deleted and only the candidate faults of S_4 are kept in the final candidate faults. If this situation does not happen, and some suspect faults of S_1, S_2, and S_4 indeed obtain 100% probability simultaneously, they are all kept as the final candidate faults except the following situation.

The considered exception is the suspect faulty net a_{32} of S_4. Because a_{32} can only arrive at D, the situation "when Q_0=0, Q=0" and "when Q_0=1, Q=1" will not happen if a_{32} occurs stuck-at fault. Nevertheless, if the actual stuck-at faulty net is a_{32}, the probability of $a_{32}/0$, $a_{32}/1$, c_k, i_k, and j_k will all be 100%. But if the actual stuck-at faulty net is c_k, i_k, or j_k, the probability of $a_{32}/0$ and $a_{32}/1$ may not be 100%. Therefore, if $a_{32}/0$, $a_{32}/1$, c_k, i_k, and j_k are selected as candidates simultaneously, c_k, i_k, and j_k are deleted and only a_{32} is kept in the final candidate faulty nets.

3.2.8 S_1, S_2 vs S_5

If some suspect faults of S_1, S_2, and S_5 indeed obtain 100% probability simultaneously, they are all kept as the final candidate faults except the following situation.

If the actual fault is a_k/T, the suspect fault a_k/T will obviously obtain 100% probability. But meanwhile, the suspect faulty nets c_1, c_2, ..., c_{k-1}, c_k, i_1, i_2, ..., i_k, i_{k+1}, j_1, j_2, ..., j_k, j_{k+1} will also obtain 100% probability. This is because in every diagnostic challenge pair of the suspect faults at $c_1{\sim}c_k$, $i_1{\sim}i_{k+1}$, $j_1{\sim}j_{k+1}$, the two challenges have the same values of $c_k{\sim}c_{32}$. The actual fault a_k/T will always at one of the two paths to D and $CLOCK$, so with the same values of $c_k{\sim}c_{32}$, the two challenges will always have the same response, which results in that all the suspect faulty nets $c_1{\sim}c_k$, $i_1{\sim}i_{k+1}$, $j_1{\sim}j_{k+1}$ have 100% probability. Nevertheless, if the actual fault occurs at one of $c_1{\sim}c_k$, $i_1{\sim}i_{k+1}$, $j_1{\sim}j_{k+1}$, the probability of a_k/T may not be 100%. Therefore, if a_k/T, $c_1{\sim}c_k$, $i_1{\sim}i_{k+1}$, $j_1{\sim}j_{k+1}$ are selected as candidates simultaneously, $c_1{\sim}c_k$, $i_1{\sim}i_{k+1}$, $j_1{\sim}j_{k+1}$ are deleted and only a_k is kept in the final candidate faulty nets. Similarly, if b_k/T, $c_1{\sim}c_k$, $i_1{\sim}i_{k+1}$, $j_1{\sim}j_{k+1}$ are selected as candidates simultaneously, $c_1{\sim}c_k$, $i_1{\sim}i_{k+1}$, $j_1{\sim}j_{k+1}$ are also deleted and only b_k is kept in the candidate faulty nets.

Notice that the delay faults at p_k, q_k, r_k, and s_k are different from a_k/T and b_k/T, because p_k/T, q_k/T, r_k/T, and s_k/T may not be at either of the two paths. Hence, $c_1{\sim}c_k$, $i_1{\sim}i_{k+1}$, $j_1{\sim}j_{k+1}$ may not obtain 100% probability when the actual fault is either of p_k/T, q_k/T, r_k/T, and s_k/T.

3.2.9 S_4 vs S_5

Due to different fault behaviors of S_4 and S_5, the situation that some suspect faults of S_4 and S_5 simultaneously obtain 100% probability is not likely to happen. If it really happens, they are all kept as the final candidate faults, except the following situation.

If the actual fault is a_{31}/T, both suspect faults a_{31}/T and $r_{32}/1$ will obtain 100% probability. This is because r_{32} can only arrive at D when c_{32} is 1. When c_{32} is 1, a_{31}/T causes Q to always be 1, so the probability of $r_{32}/1$ is 100%. Nevertheless, if the actual fault is $r_{32}/1$, the suspect fault a_{31}/T may not obtain 100% probability. Therefore, if a_{31}/T and $r_{32}/1$ are selected as candidate faults simultaneously, $r_{32}/1$ is deleted and only a_{31}/T is kept in the final candidate faulty nets. Similarly, if b_{31}/T and $p_{32}/1$ are selected as candidate faults simultaneously, $p_{32}/1$ is deleted and only b_{31}/T is kept in the final candidate faulty nets.

In addition, we also find some indistinguishable faults: $p_{32}/1$ and s_{32}/T, $r_{32}/1$ and q_{32}/T, $a_{32}/1$ and b_{32}/T. For example, when c_{32}=0, both $p_{32}/1$ and s_{32}/T make Q always be 1; when c_{32}=1, both $p_{32}/1$ and s_{32}/T are bypassed. Hence $p_{32}/1$ and s_{32}/T are indistinguishable.

3.2.10 S_1, S_2, S_4, S_5 vs S_3

Due to different fault behaviors of S_3 from S_1, S_2, S_4, and S_5, if t is selected as the candidate net, it is not likely that suspect faults of S_1, S_2, S_4, and S_5 can also obtain 100% probability. If it really happens, they are all kept as the final candidate faults.

In addition, $b_{32}/0$ and $b_{32}/1$ are indistinguishable from $t/0$ and $t/1$. Since b_{32} can only arrive at $CLOCK$, the stuck-at fault behavior of b_{32} is totally the same as t.

4. Experimental Results

In the experiments, we simulate 100 arbiter PUFs with 16-bit challenge, 100 arbiter PUFs with 32-bit challenge, and 100 arbiter PUFs with 64-bit challenge. For each arbiter PUF, the delay variations of p_k, q_k, r_k, s_k, a_k, and b_k are assumed obeying Gaussian distribution.

We generate N_{CP} diagnostic challenge pairs for each suspect faulty net c_k of S_1, and generate N_{CP} diagnostic challenge pairs for each suspect fault $i_k/0$ and $i_k/1$ of S_2. These $4{\times}N_{CP}$ diagnostic challenges are also used for distinguishing S_3, S_4, and S_5. We find they are sufficient, so no more diagnostic challenges are specifically generated for S_3, S_4, and S_5.

For each arbiter PUF, we inject every one fault of $S_1{\sim}S_5$ into it to form a failing arbiter PUF instance. Hence if the arbiter PUF contains N_S suspect faults, N_S failing arbiter PUF instances are formed. Totally 270000 failing arbiter PUF instances are formed, each of which is diagnosed by the proposed method to obtain the candidate faulty nets. The reported candidate faulty nets are finally evaluated using the two metrics: diagnostic accuracy and diagnostic resolution. The diagnostic accuracy is 100% if the candidate faulty nets indeed contain the actual faulty net. The diagnostic resolution is the number of candidate faulty nets. The ideal value of diagnostic resolution is 1.

The experimental results of the average diagnostic accuracy and diagnostic resolution are shown in Table I. With increasing of N_{CP}, both diagnostic accuracy and diagnostic resolution are improved. This is because our diagnosis method is based on the uniformity of arbiter PUF. When N_{CP} is 100, the average diagnostic accuracy achieves 100.0% and the average diagnostic resolution achieves 1.7.

Fig.9 shows the distribution of diagnostic resolution when N_{CP} is 100. On average, 65% fault diagnosis instances report only one candidate faulty net which is exactly the actual faulty net. Some fault diagnosis instances report more than one candidate faulty nets. Most of these instances are due to the inherent indistinguishability as explained in Section 3. For example, $i_k/0$ and $j_k/0$ are indistinguishable. To further distinguish them, design for diagnosis may be needed. For other instances, using more diagnostic challenges may be useful.

5. Conclusion

This paper proposes a fault diagnosis method toward the classic arbiter PUF. The stuck-at faults and the delay faults within arbiter PUF are considered. All the suspect faults are classified into five sets. A diagnostic challenge generation method and a corresponding CRP analysis method are proposed to firstly distinguish the suspect faults belong to the same set, and then to distinguish the suspect faults belong to different sets. Experimental results show 100.0% diagnostic accuracy and good resolution can be achieved.

6. Reference

[1] U. Ruhrmair, D. E. Holcomb, "PUFs at a Glance," DATE, 2014.
[2] P. Tuyls, G. J. Schrijen, et. al., "Read-Proof Hardware from Protective Coatings," CHES, pp. 369-383, 2006.
[3] J. Guajardo, S. S. Kumar, et. al., "FPGA Intrinsic PUFs and Their Use for IP Protection," CHES, pp. 63-80, 2007.
[4] P. Koeberl, U. Kocabas, A.-R. Sadeghi, "Memristor PUFs: A New Generation of Memory based Physical Unclonable Functions," DATE, pp. 428-431, 2013.
[5] M. S. Hashemian, B. Singh, et. al., "A Robust Authentication Methodology using Physically Unclonable Functions in DRAM Arrays," DATE, pp. 647-652, 2015.
[6] D. Lim, "Extracting Secret Keys from Integrated Circuits," MSc Thesis, MIT, 2004.
[7] G. E. Suh, S. Devadas, "Physical Unclonable Functions for Device Authentication and Secret Key Generation," DAC, pp. 9-14, 2007.
[8] M. Majzoobi, F. Koushanfar, M. Potkonjak, "Lightweight Secure PUFs," ICCAD, pp. 670-673, 2008.
[9] J. Ye, Y. Hu, X. Li, "OPUF: Obfuscation Logic Based Physical Unclonable Function," IOLTS, pp. 156-161, 2015.
[10] M. Majzoobi, F. Koushanfar, M. Potkonjak, "Testing Techniques for Hardware Security," ITC, Paper 31.3, 2008.
[11] S. U. Hussain, M. Majzoobi, F. Koushanfar, "BIST-PUF: Online, Hardware-based Evaluation of Physically Unclonable Circuit Identifiers," ICCAD, pp. 162-169, 2014.
[12] I. Pomeranz, S. M. Reddy, "Selection of a Fault Model for Fault Diagnosis Based on Unique Responses," DATE, pp. 994-999, 2009.
[13] W.-T. Cheng, Y. Huang, "Enhance Profiling-Based Scan Chain Diagnosis by Pattern Masking," ATS, pp. 255-260, 2010.
[14] Y. Zhang, V. D. Agrawal, "A Diagnostic Test Generation System," ITC, paper 12.3, 2010.
[15] R. D. Blanton, W. C. Tam, et. al., "Yield Learning Through Physically Aware Diagnosis of IC-Failure Populations," IEEE Design & Test of Computers, 29(1), pp. 36-47, 2012.
[16] P.-Y. Hsuch, S.-F. Kuo, et. al., "Case Study of Yield Learning Through In-House Flow of Volume Diagnosis," VLSI-DAT, pp. 1-4, 2013.
[17] J. Ye, Y. Huang, et.al., " Diagnosis and Layout Aware (DLA) Scan Chain Stitching," IEEE TVLSI, vol. 23, no. 3, pp. 466-479, 2015.
[18] J. Ye, Y. Hu, X. L, "Diagnosis of Multiple Arbitrary Faults with Mask and Reinforcement Effect," DATE, pp. 885-890, 2010.
[19] M. Rostami, M. Majzoobi, etl. al., "Robust and Reverse-Engineering Resilient PUF Authentication and Keyexchange by Substring Matching," IEEE TETC, vol. 2, no. 1, pp. 37-49, 2014.

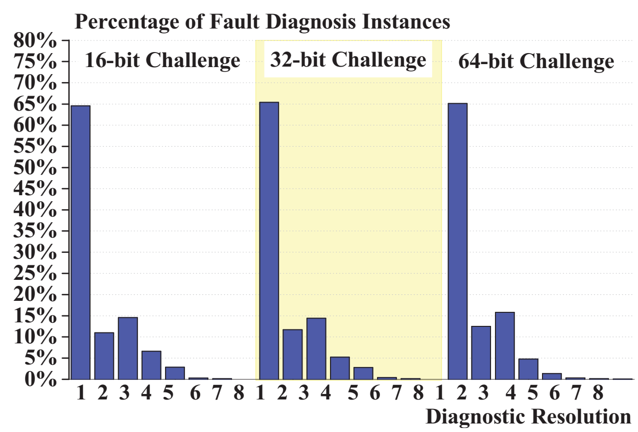

Fig.9 Distribution of Diagnostic Resolution

Table I Experimental Results of Diagnostic Accuracy and Diagnostic Resolution

Number of Challenge Bits	N_S	N_{CP}	Total Number of Diagnostic Challenges	Average Diagnostic Accuracy	Average Diagnostic Resolution
16	388	20	1920	99.6%	3.09
		40	3840	99.8%	2.26
		60	5760	99.8%	1.87
		80	7680	99.9%	1.84
		100	9600	100.0%	1.74
32	772	20	3840	99.7%	4.31
		40	7680	99.9%	2.27
		60	11520	99.9%	1.92
		80	15360	99.9%	1.75
		100	19200	100.0%	1.70
64	1540	20	7680	99.8%	6.78
		40	15360	99.9%	3.40
		60	23040	99.9%	2.09
		80	30720	99.9%	1.72
		100	38400	100.0%	1.66