TimingCamouflage+: Netlist Security Enhancement With Unconventional Timing

Grace Li Zhang, Bing Li, Meng Li, *Member, IEEE*, Bei Yu, David Z. Pan, *Fellow, IEEE*, Michaela Brunner, Georg Sigl, and Ulf Schlichtmann, *Senior Member, IEEE*

***Abstract*—With recent advances in reverse engineering, attackers can reconstruct a netlist to counterfeit chips by opening the die and scanning all layers of authentic chips. This relatively easy counterfeiting is made possible by the use of the standard simple clocking scheme, where all combinational blocks function within one clock period, so that a netlist of combinational logic gates and flip-flops is sufficient to duplicate a design. In this article, we propose to invalidate the assumption that a netlist completely represents the function of a circuit with unconventional timing. With the introduced wave-pipelining (WP) paths, attackers have to capture gate and interconnect delays during reverse engineering, or to test a huge number of combinational paths to identify the WP paths. To hinder the test-based attack, we construct false paths with WP to increase the counterfeiting challenge. The experimental results confirm that WP true paths and false paths can be constructed in benchmark circuits successfully with only a negligible cost, thus thwarting the potential attack techniques.**

***Index Terms*—Anti-counterfeiting, embedded security, IC camouflage, netlist camouflage, netlist security, reverse engineering, wave pipelining.**

I. Introduction

A MAJOR integrated circuit (IC) counterfeiting threat is the illegal production of chips by a third party with a netlist reverse engineered from authentic chips. In reverse engineering, authentic chips are delayered and imaged to identify logic gates, flip-flops, and their connections to reconstruct a netlist. Because the recognized netlist carries all necessary design information, this reverse engineering flow allows counterfeiters to reproduce authentic chips with much freedom.

Several techniques have been proposed to thwart reverse engineering attacks on authentic chips. The first method is IC camouflaging which tries to prevent the netlist from being recognized easily [2]. In [3], transistors are manipulated with a stealthy doping technique during manufacturing so that they function differently than they appear. In [4], the layouts of different cells are designed to be identical, leading to difficulty in interpreting the functionality of a netlist. The work in [5]–[7] mixes real and dummy contacts to camouflage standard cells so that they cannot be recognized by reverse engineering. The method in [8] explores netlist obfuscation by iterative logic fanin cone analysis at the circuit level. In addition, the method in [9] introduces a quantitative security criterion and proposes camouflaging techniques with a low-overhead cell library and an AND-tree structure to strengthen netlist security. The functionality of a given design is perturbed in [10] by applying a simple transformation and a separate camouflaged block is used to recover the functionality. Moreover, the method in [11] creates logic loops by adding dummy wires and gates to obfuscate the circuit topology. However, nearly all these camouflaging methods can be deobfuscated by attacks based on Boolean satisfiability (SAT) [12]–[14].

The second method to thwart reverse engineering is logic locking, which inserts additional logic gates, e.g., XOR/XNOR in [15] and [16], AND/OR in [17], MUX in [18], and look-up tables (LUTs) in [19], into the netlist. These components can only activate the correct function of the circuit with a given key. This method is expanded in [20] to incorporate delay information into the locking mechanism. Furthermore, logic locking can be performed at sequential level to prevent the circuit from entering working states without a valid key [21]. Recently, logic locking has been applied to protect parametric behavior of circuits, e.g., pipelined processors [22], GPUs [23], and analog circuits [24]. The method in [22] adds meaningless clock cycles to camouflage a design, where only correct keys allow a high timing performance. However, various attack methods, e.g., SAT attack [25], removal attack [26], and bypass attack [26], [27] can potentially recognize the correct keys.

Beyond the techniques described above, other methods, e.g., watermarking [28], metering [29], and split manufacturing [30], [31], can also be applied against counterfeiting. But the purpose of these methods is primarily to prevent

Manuscript received March 30, 2019; revised June 14, 2019, September 2, 2019, and December 26, 2019; accepted January 26, 2020. Date of publication February 17, 2020; date of current version November 20, 2020. This work was supported in part by the Fraunhofer High Performance Center Secure Connected Systems Munich. A preliminary version of this article was published in the Proceedings of the Design, Automation, and Test in Europe (DATE) Conference, 2018 [1]. This article was recommended by Associate Editor R. Karri. *(Corresponding author: Bing Li.)*

Grace Li Zhang, Bing Li, and Ulf Schlichtmann are with the Chair of Electronic Design Automation, Technical University of Munich, 80333 Munich, Germany (e-mail: grace-li.zhang@tum.de; b.li@tum.de; ulf.schlichtmann@tum.de).

Meng Li is with Facebook, Menlo Park, CA, USA (e-mail: meng_li@utexas.edu).

David Z. Pan is with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712 USA (e-mail: dpan@ece.utexas.edu).

Bei Yu is with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong (e-mail: byu@cse.cuhk.edu.hk).

Michaela Brunner and Georg Sigl are with the Chair of Security in Information Technology, Technical University of Munich, 80333 Munich, Germany (e-mail: michaela.brunner@tum.de; sigl@tum.de).

Digital Object Identifier 10.1109/TCAD.2020.2974338

overproduction and they need to be adapted properly to counter reverse engineering.

The existing methods of circuit protection either make the netlist more difficult to be recognized, or make the correct behavior of the circuit dependent on additional input information even after the netlist is recognized. In this article, we propose a new perspective to counter counterfeiting based on reverse engineering. Our contributions are as follows.

1) A new dimension of IC camouflage is proposed to secure circuit netlists by integrating unconventional timing information. A camouflaged netlist thus only works correctly with a given set of timing information, which, however, is difficult to be recognized exactly by reverse engineering.
2) To integrate unconventional timing into a netlist, wave-pipelining (WP) paths are constructed in some parts of a circuit. To prevent the exposure of these paths resulting from clustering, the retiming technique is deployed to spread them across a circuit by blocking the paths not related to WP construction.
3) With the retiming technique, the area overhead to construct WP paths can also be reduced, since paths not related to WP construction are maintained as single-period to avoid unnecessary delay insertion.
4) A camouflaged netlist with WP only contains normal logic gates, so that it is challenging for attackers to isolate and then identify the locations of WP. An attack method based on the path delay test to locate the camouflaged WP paths would require a large number of test vectors.
5) The introduced WP false paths obstruct the test-based counterfeiting methods further, because some paths that are originally testable are camouflaged as false paths in the netlist.
6) The proposed method is fully compatible with other security techniques introduced previously, so that they can be combined together to strengthen netlist security at the circuit level.

The remainder of this article is organized as follows. In Section II, we explain the motivation and the basic idea of the proposed method. In Section III, we provide a detailed description of the WP technique. In Section IV, we analyze potential attack techniques to identify or circumvent WP paths introduced into the circuit. We also discuss the limitations of these techniques and propose countermeasures to thwart the attack attempts. We describe the implementation details to construct WP paths in Section V. The experimental results are reported in Section VI. Conclusions are drawn in Section VII.

II. Motivation and Basic Concept

Digital circuits rely on their structures to define their functions. A netlist is usually sufficient to reproduce a correctly working circuit. To prevent the netlist from being recognized by reverse engineering, techniques from the physical level to netlist level can be applied to camouflage the logic. These methods, however, are still confined within the conventional single-period clocking timing model, so that attackers only need to recognize the netlist correctly.

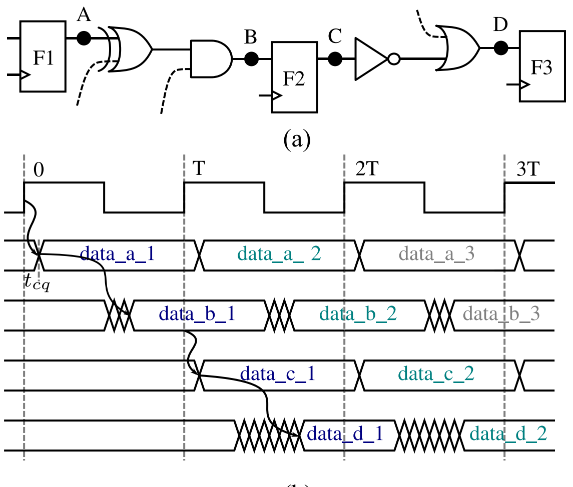


Fig. 1. Conventional timing. (a) Part of a sequential circuit. (b) Single-period clocking.

In the conventional clocking timing model, all the paths in a combinational block operate within one clock period. We call them single-period clocking paths. Fig. 1(a) shows an example of a conventional sequential circuit with three flip-flops F1, F2, and F3. The data switching activities at internal points A, B, C, and D in this circuit are illustrated in Fig. 1(b). We assume that data are latched into flip-flops at the rising clock edge. At time 0, the input data of F1 is transferred to its output and becomes stable after t_{cq}, the clock-to-q delay of F1, shown as data_a_1. It travels further through the logic gates and reaches B, shown as data_b_1 in Fig. 1(b). Although the data at B is stable far before the next rising clock at time T, it is still blocked at F2 until the arrival of the next rising clock edge to be transferred to the output of F2. At this clock edge, the second data wave is injected onto the path from A to B by F1 and starts to propagate. In this way, combinational logic blocks are isolated by flip-flops and data waves are pipelined to propagate through the logic blocks in the conventional digital design.

A side effect of the sequential isolation with flip-flops above is that the netlist carries all logic information. This simplification allows attackers to counterfeit chips relatively easily, because they only need to recognize the logic types of gates, flip-flops, and interconnect connections with reverse engineering.

To thwart the potential netlist attack attempt described above, we propose to invalidate the conventional timing model in the circuit under protection. For example, we can remove the flip-flop in the middle of Fig. 1(a) to construct the circuit structure shown in Fig. 2(a). The switching activities of the internal signals in Fig. 2(a) are illustrated in Fig. 2(b). At two consecutive rising clock edges, F1 injects data_a_1 and data_a_2 onto the combinational path, respectively. Therefore, two data waves at A are always separated by one clock period. Since no flip-flop blocks the propagation of data_b_1, it passes through C directly and reaches D after traveling through the

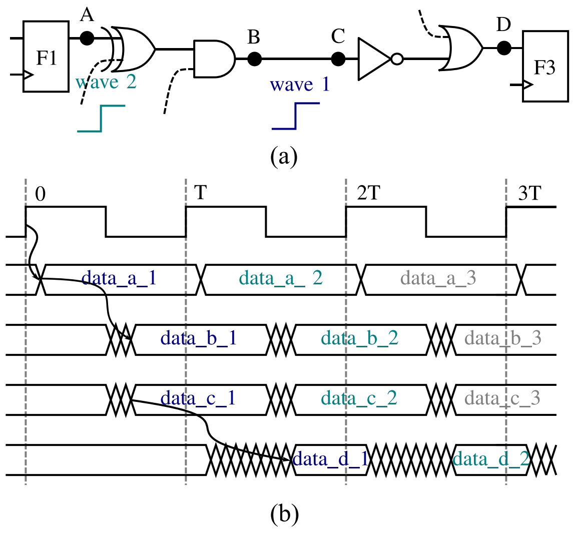


Fig. 2. WP. (a) Sequential circuit after F2 is removed. (b) Pipelining with two data waves.

inverter and OR gate. Once the data at D becomes stable as data_d_1, it waits to be latched by F3 while the second data wave is propagating. To avoid that data_d_1 is flushed by the following data waves, the delays of all the combinational paths passing through B and C, including those between F1 and F3, must be larger than one clock period. Otherwise, the change of the data at D triggered by the second data wave data_d_2 happens before the next rising clock edge, so that the previous data_d_1 waiting at the input of F3 cannot be latched correctly. The result of flip-flop removal is that two data waves propagate along the path without a flip-flop separating them. This technique is called WP and has previously been investigated for circuit optimization as in, e.g., [32]–[35].

With WP paths, the function of the circuit depends on both its structure and the timing information of combinational paths. If attackers obtain a netlist as in Fig. 2(a), they need to determine whether these paths are single-period clocking paths or WP paths. If attackers assume the former and process the netlist using a standard EDA flow, the circuit loses synchronization, because the data at the input of F3 is latched one clock period earlier than in the original design. If attackers want to determine whether it is the latter case, additional effort is required to extract the timing information for each combinational path in the circuit.

Though attackers may have access to the standard cell library, e.g., through a third-party IP vendor, it is still very hard to obtain accurate interconnect/RC parasitics by delayering authentic chips, due to unknown process parameters, challenges in 3-D RC extraction, and switching-window-dependent crosstalk-induced delay variations, etc. In any case, the more accurate the original timing information should be recognized from delayered chips, the harder and more expensive it becomes to reproduce a design. Therefore, this unconventional timing concept has a potential to open up a new dimension of netlist security and can be combined with the existing camouflaging techniques, e.g., through dopant-level camouflage [3] or dummy contact insertion [5]–[7].

III. Wave-Pipelining Constraints

A WP path such as the one in Fig. 2(a) allows two data waves to propagate on the path simultaneously. Since the second data wave must not catch the first one, special timing constraints should be imposed for this path.

When creating WP paths, a flip-flop in Fig. 1(a) is removed to construct the circuit in Fig. 2(a). In practice, this operation may lead to many paths with wave pipelining, because any combinational path through B and C becomes a new WP path. When the setup time t_{su} and the hold time t_h of the flip-flop are considered, all these paths P should meet two constraints. First, the delay d_p of a path should be t_h larger than the clock period T. Otherwise, the second wave arrives at the flip-flop too early and thus disturbs the latching process of the first wave. Second, the delay of the path should be t_{su} smaller than $2T$ to guarantee that the data is latched by F3 correctly. The timing constraints for all these paths can be written as

$$d_p \geq T + t_h \quad \forall p \in P \iff \min_{p\in P}\{d_p - t_h\} \geq T \tag{1}$$

$$d_p \leq 2T - t_{su} \quad \forall p \in P \iff \max_{p\in P}\{d_p + t_{su}\} \leq 2T. \tag{2}$$

If all the WP paths meet the two constraints (1) and (2), the WP version of the circuit after a flip-flop is removed is functionally equivalent to the original circuit.

IV. Potential Attacks and Countermeasures

The proposed camouflage technique with WP secures netlists with timing information at sequential level. This new technique may face potential attacks. We analyze some of these attacks in this section, though their experimental attempt is not covered in this article. In the assumed attack model, the available information includes a netlist recognized by reverse engineering and estimated delays of logic gates as well as interconnects with an inaccuracy factor τ. The objective of the attack is to identify on which combinational paths in the netlist WP is applied.

1) First Attack Technique—Delay Estimation: In this method, the delays of all the gates and interconnects are measured while the netlist is reverse engineered. Using the measured delays, path delays can be estimated from the netlist. Since the delays of WP paths are between T and $2T$ as defined in (1) and (2), these paths can therefore be identified. The challenge of this attack technique is that it is difficult to extract accurate gate and interconnect delays just from reverse engineering, due to the inaccuracy in delayering authentic chips described in Section II. Assume that the real delay of a path is d, including setup time of a flip-flop, and the delay recognition technique suffers an inaccuracy factor τ ($0 < \tau < 1$). Consequently, this path delay can be any value in the range $[(1-\tau)d, (1+\tau)d]$. If the upper bound of an estimated delay is smaller than T, this path is definitely a single-period clocking path. If the lower bound of an estimated delay is larger than T, the path is definitely a WP path. However, if no such clear decision can be made with the estimated delay, namely,

$$(1-\tau)d \leq T \leq (1+\tau)d \tag{3}$$

this path can only be considered as suspicious of WP. In the following, we call the range $[(1-\tau)d, (1+\tau)d]$ the *gray region* for a path with delay d. In reality, a well-optimized design contains a huge number of critical paths with delays close to the clock period T, so that their gray regions often surround T. When constructing WP paths in the proposed method, we also guarantee that their delays are in the gray region to counter this attack technique.

2) Second Attack Technique—Testing Delays: With the estimated delays, attackers can actually narrow down the number of potential WP paths, because paths with estimated delays definitely smaller or larger than T can be screened out. The second attack technique is thus to test the delays of the remaining suspicious paths using authentic chips from the market. With the netlist recognized, it is not difficult to determine test vectors to trigger the remaining suspicious paths. Since the only information of interest is whether a path delay is larger than T, only one delay test for each path is required.

To prevent all suspicious paths from being tested as described above, we introduce a countermeasure to create unsensitizable paths with WP. When we construct WP paths by removing flip-flops, we prefer the paths that, viewed directly with the conventional single-period clocking model, are false paths, which cannot be sensitized by any test vector.

Definition 1 (False Path): A combinational path which cannot be activated in functional mode or tested due to controlling signals from other paths [36], [37].

Definition 2 [Wave-Pipelining False Path (WP False Path)]: A combinational path with wave pipelining that is a false path when viewed with the conventional single-period clocking model.

Definition 3 [Wave-Pipelining True Path (WP True Path)]: A combinational path with wave pipelining that is a true path when viewed with the conventional single-period clocking model.

WP false paths have two data waves propagating along them when the circuit is operating, but they are false paths when the netlist is examined assuming the conventional single-period clocking model. An example of a WP false path is shown in Fig. 3, which is a snippet of the s298 circuit from the ISCAS89 benchmark set. When the flip-flop in the middle is removed, the dashed path becomes a WP path. If attackers view it as a single-period clocking path in the extracted netlist, this dashed path is also a false path. In this case, a signal switching at the beginning of the dashed path never reaches the final flip-flop. If the signal v_2 has a value "1," which is the controlling signal to an OR gate, it blocks the signal switching along the dashed path at the last OR gate; If the signal v_2 has a value "0," it blocks the signal switching along the dashed path at the AND gate right away. Consequently, the dashed path cannot be triggered for delay test and attackers have no way to differentiate it from all the other false paths in the original circuit, which may contribute up to 75% of all the combinational paths in real circuits [38].

3) Third Attack Technique—Logic Simulation: Since the delays of false paths cannot be tested, brute-force logic simulation could be applied to differentiate the camouflaged false paths from real false paths. In this method, each false path

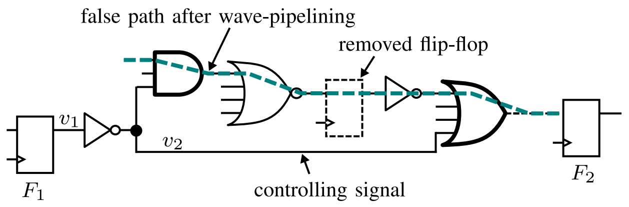


Fig. 3. Two true paths form a WP false path.

that cannot be excluded by delay screening in the first step can be assumed to be a real false path or a WP false path, so that attackers have to verify which assumption is correct for this false path with simulations. Assuming the number of such paths is n. If each path in n can be a real false path or a WP false path, then 2^n simulations of the complete circuit have to be performed to check which combination is correct. In theory, this method can eventually find the correct combination of real false paths and WP false paths. However, it is still impractical because of the unaffordable simulation time due to the large number of false paths in the original design [36], [38], e.g., 728 262 for s13207 in the ISCAS89 benchmark set, and the long runtime for a full simulation of the complete circuit.

4) Fourth Attack Technique—Sizing False Paths: In this method, all false paths in the circuit are considered as WP paths and logic gates are sized so that delays of all these paths meet constraints (1) and (2). The concept behind this technique is that false paths are not triggered anyway so that they do not affect the logic of the circuit if their delays are larger than the clock period T. This assumption, however, is incorrect because false paths sized to have delays larger than T may still affect the normal circuit operation. Another challenge of this attack technique is that it is very difficult to find a solution to size a huge number of false paths without affecting the normal true paths whose delays should be smaller than T.

5) Fifth Attack Technique—Delay Calculation: In this method, all gate and interconnect delays in a circuit are calculated from path delays measured by testing. Since path delays are linear combinations of the gate and interconnect delays, the measured path delays can be used to calculate gate and interconnect delays by linear algebra. This method needs to deal with several new challenges. First, in a commercial design, a large number of combinational paths will be tested. Second, all logic gates and interconnect segments should appear on testable paths in a way that the coefficient matrix of the system of linear equations has a rank equal to the number of gates and interconnect segments, even in view of a large percentage of false paths. Third, inaccuracy in at-speed test of path delays exists due to environmental factors such as noise and temperature as well as the nature of binary-search of at-speed delay test.

6) Sixth Attack Technique—Pattern Recognition: To construct WP paths such as the dashed path in Fig. 3, the flip-flop in the middle is removed, leading to a relatively large number of logic gates along this path. In addition, interconnects might be lengthened to enlarge the path delay to satisfy the timing constraints (1) and (2). The special patterns of long interconnects and gate chains might expose the locations of

WP paths. The sixth attack technique is to search such special patterns to identify the WP paths. However, this attack method faces many challenges. First, the uncritical single-period paths have smaller delays compared with the critical paths. To guarantee the timing constraints of the critical paths, interconnects along them are usually short. On the contrary, the length of interconnects along uncritical paths might be large. Consequently, whether WP exists cannot be determined simply by the length of interconnects. Second, the delays of the logic gates can be camouflaged with multithreshold voltage (MVT) technique [39]–[41], where high V_t gates have larger delays than low V_t gates. From the view of the layout, there is no difference between both types of gates, because the modulation of the threshold voltages is achieved by changing channel doping concentration during manufacturing [42]. To reduce the number of logic gates along WP paths, the logic gates can be replaced with high V_t counterparts to enlarge their delays. Consequently, attackers cannot determine the locations of WP paths according to the number of logic gates.

7) Seventh Attack Technique—SAT-Based Attack: In the SAT-based attack methods, e.g., [14] and [43], it is assumed that attackers have full access to the scan chain, so that they can apply input test vectors and observe the outputs with the authentic chips. With various sets of input and output observations, the SAT-based attack can determine the locations of WP paths that match all input and output observations. In fact, WP true paths can be screened out with this method, since these paths can be triggered with the at-speed testing. However, the constructed WP false paths cannot be triggered for delay test if they are considered to work within one clock period. If attackers try to activate such paths with two consecutive data waves, all the side-inputs of the WP false paths should be set to noncontrolling values in two consecutive clock cycles. For example, v_2 in Fig. 3 is one of the side-inputs of the dashed WP false path. It should be set to 1 in the first clock cycle to allow a signal switching through the AND gate. In the second clock cycle, it should be set to 0 so that the signal switching can pass through the OR gate. These requirements might be met by tracing logic blocks between two flip-flop stages before F1. The traced logic blocks should be set to appropriate values so that all the side-inputs ensure the activation of the WP false paths. However, this method requires drastic changes in the existing testing platform. In addition, the delays of original false paths in the circuits might be larger than the given clock period since they are ignored during timing analysis. By triggering the conflicting logic with two clock cycles, these paths can also be activated with two consecutive data waves, so that the constructed WP false paths can be concealed. Furthermore, TimingCamouflage+ can be combined with other existing methods such as scan chain encryption [44], [45], to further increase the difficulty of activating WP paths.

To differentiate WP false paths from original false paths with SAT-based attacks, the whole circuit can be considered as a black box, where only the data at the primary inputs and the primary outputs of the design can be observed. Since identifying WP false paths requires to determine where the flip-flops are removed, attackers can first collect connections between gates along all suspicious false paths and then determine where to reinsert flip-flops to recover the original circuit without WP. To identify the correct combinations of inserting flip-flops, SAT-based attacks search iteratively for discriminating input sequences at the primary inputs. Each discriminating input sequence eliminates one or more combinations of inserting flip-flops. The iteration continues until only the correct combination of inserting flip-flops remains. In [46] and [47], similar attacks have been attempted onto sequential circuits, and it has been demonstrated that it is not possible to decamouflage relatively large sequential circuits even with smaller numbers of keys compared with TimingCamouflage+. Furthermore, TimingCamouflage+ actually introduces a new dimension in netlist camouflaging. Therefore, it can also be combined with other security methods, e.g., Anti-SAT logic locking [48], [49], to counter SAT-based attacks together.

Recently, a SAT modulo theory (SMT)-based attack method is proposed to enhance SAT-based attack with theory and graph solvers [50]. However, it is assumed that all combinational paths work within one clock period. This assumption does not hold in TimingCamouflage+, where the intentionally constructed WP paths have delays larger than one clock period. To extract correct timing constraints of such WP paths, their locations have to be identified, which requires much effort as discussed above.

V. Wave-Pipelining Construction

When constructing WP paths in a circuit while maintaining its original function, we need to guarantee that the constructed paths meet the timing constraints (1) and (2). To counter the attack techniques discussed in Section IV, the constructed paths should meet constraint (3), so that they cannot be verified easily. Furthermore, the WP paths should contain false paths when viewed as single-period clocking paths. The WP construction problem can thus be formulated as follows.

Inputs: Original optimized design; gate and interconnect delays; the given clock period T; the delay recognition inaccuracy factor τ $(0 < \tau < 1)$; the required number of WP false and true paths n_{wpf}, n_{wpt}; and distance threshold dist.

Outputs: A revised design containing at least n_{wpf} WP false paths and n_{wpt} WP true paths, where n_{wpf} and n_{wpt} are user-defined parameters. These WP paths should meet the timing constraints (1) and (2) as well as the gray region requirement (3).

Objectives: The original function of the circuit should be maintained; the original design should be kept unchanged as much as possible; the increased resource usage should be as little as possible; the physical distances between the constructed WP paths should be as far as possible to prevent the exposure of these paths resulting from clustering.

When constructing WP paths, we incorporate process, voltage and temperature (PVT) variations by allowing path delays to deviate from their original values. Specifically, delays of the longest paths are enlarged to $(1 + \delta)$ times of the original values. On the contrary, delays of shortest paths are reduced to $(1 - \delta)$ times of the original values. The value of δ should be determined by designers according to the corresponding

```
Input: netlist, delay information, $T$, $\tau$, $n_{wpf}$, $n_{wpt}$, $dis_t$;          L1
$F_w = \varnothing$;  // the flip-flops that cannot be used for construction
                                   ↓
Sort all flip-flops F in a decreasing order;                                       L3
Filter F $\rightarrow$ $F_n$ using the number of source/sink flip-flops;          L4
For i=1 to $|F_n|$do                                                               L5
    If a flip-flop $ff_i \notin F_w$ then
        $n_f$=check_WP_false_paths($ff_i$, $T$, $\tau$);                          L7
        If $n_f > 0$ then
            If construct_WP_paths($ff_i$, $T$, $\tau$)=false, go to L5;            L9
            $F_w \leftarrow ff_i$, fanin($ff_i$) and fanout($ff_i$);              L10
            If($DIS(ff_i, ff_j) < dis_t$) //$ff_j$ are other flip-flops
                $F_w \leftarrow ff_j$;                                            L12
        $n_{wpf} = n_{wpf} - n_f$;                                                L13
        If $n_{wpf} \leq 0$ then                                                   L14
        break;                                                                     L15
                                   ↓
Construct wave-pipelining true paths similar to L5–L15
```

Fig. 4. Major steps of WP construction.

manufacturing technology. With this setting, the constructed WP paths should still work correctly under PVT variations.

A. Work Flow of Wave-Pipelining Construction

The major steps to construct WP paths are shown in Fig. 4. WP paths can potentially be constructed at a flip-flop connected to paths with large delays, so that timing constraints (1) and (2) of such paths can be satisfied easily. Therefore, we sort all flip-flops in a circuit in a decreasing order according to the sum of the maximum delays of their incoming and outgoing paths as described above. In addition, WP construction might be challenging at those flip-flops with a large number of incoming and outgoing paths, because a huge number of WP paths can appear when the flip-flop is removed. To guarantee all these WP paths to meet the timing constraints (1) and (2) is difficult, because these constraints require that all the path delays should be within the range of T and 2T simultaneously. Therefore, such flip-flops with large numbers of incoming and outgoing paths should be filtered out. Since traversing all incoming and outgoing paths of a flip-flop to acquire their numbers is time-consuming, we use the numbers of source and sink flip-flops of a flip-flop to indicate the difficulty in constructing WP paths. Therefore, flip-flops with the number of source or sink flip-flops larger than a given a threshold are filtered out to accelerate the construction.

After sorting and filtering flip-flops, for each remaining flip-flop ff_i, we check whether there are WP false paths that can be formed from single-period true paths on the left and on the right of ff_i (L7). The number of such paths is stored in n_f. If WP false paths can be formed at ff_i, the function construct_WP_paths(ff_i, T, τ) is used to construct such paths eventually with the combinational logic leaving from and arriving at it. The identified logic gates are also expanded to include all the gates reachable from them, because sizing the logic gates on the WP paths also affects delays of paths through the expanded gates. All these gates are denoted together as a set G. The details of this construction will be explained later.

As shown in Fig. 2(a), for a WP path, the flip-flop at the beginning of the path and the flip-flop at the end of the path should not be removed from the circuit during constructing of further WP paths. Otherwise, paths with more than two waves may appear, requiring more complex timing constraints. These fanin and fanout flip-flops are inserted into the set F_w (L10) and all the flip-flops tracked by F_w cannot be considered as candidates to construct WP paths. In addition, the physical distance between the WP paths should be large to prevent the exposure resulting from the clustering of such paths. We use the distances between flip-flops to represent the distances between WP paths. Therefore, a flip-flop ff_j whose distance to ff_i that is currently used to construct WP is smaller than dist cannot be a candidate for WP, so that it is inserted into the set F_w.

In the last step of the proposed method, we construct WP paths that are still true viewed with the single-period clocking model. These paths are used to guarantee that attackers must test all single-period clocking and WP true paths whose delays meet the gray region requirement. Without these paths, attackers can assume all testable paths are single-period and avoid the expensive test procedure. Different from the construction of WP false paths, the construction of these true paths only relies on path delays. The path construction in this step is similar to L5–L15 in Fig. 4. The only differences in this construction are that at L7 we check WP true paths and in L13 and L14 we use n_{wpt} as the number of such paths to be constructed.

B. False Path Checking

In the work flow above, we need to check very often whether a path is false or not. In the proposed method, we only consider the statically unsensitizable paths as false paths [51], [52], such as the false path shown in Fig. 3. In this example, the path cannot be sensitized because the controlling signal v_2 blocks either the AND gate or the last OR gate, no matter what its value is. Besides statically unsensitizable paths, there are also dynamically unsensitizable paths [36], which, however, might still be sensitized by test vectors set through the scan chain [53]. Therefore, statically unsensitizable paths are more conservative in thwarting attacks and thus used in our method.

To verify whether a path is statically unsensitizable, we assign Boolean variables to the inputs and output of each gate and formulate false path checking as a SAT problem [52]. The logic relations between these variables are established according to functions of the corresponding logic gates. If a path can be sensitized, all the side inputs of the path must be set to the noncontrolling values. For example, the path in Fig. 3 requires that the condition $(v_2 \wedge \neg v_2)$ is true, which is, however, always false.

In implementing the function check_WP_false_paths(ff_i, T, τ) in Fig. 4, we randomly select 500 paths that drive the current flip-flop ff_i and exclude the false paths from them, because the WP paths to be constructed should be formed by two single-period clocking true paths. Similarly, we select 500 paths that are driven by ff_i and exclude the false paths. By limiting the number of paths we avoid to enumerate all paths while searching WP false paths, which are in fact abundant

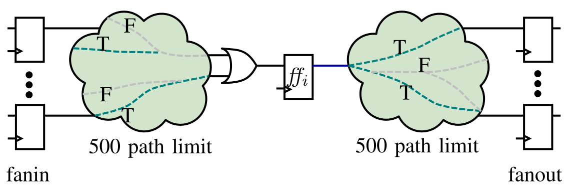


Fig. 5. Number of paths on each side of ff_i is limited to 500.

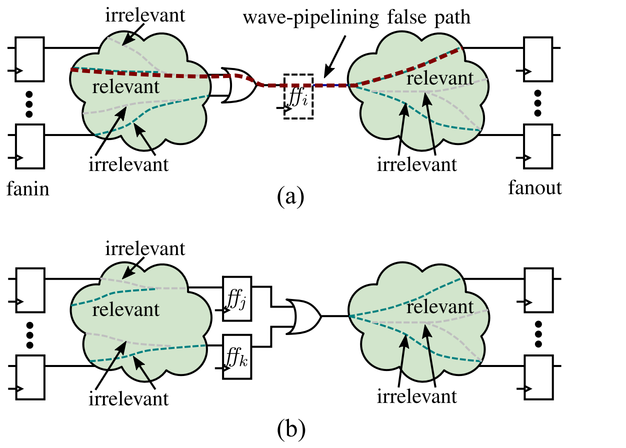


Fig. 6. Removal and retiming of a flip-flop. (a) Generation of a WP false path with the relevant paths and unnecessary WP paths with the irrelevant paths after the flip-flop ff_i is removed. (b) ff_i is retimed to the left of the OR gate to block irrelevant paths.

in the circuits as demonstrated by experimental results in Section VI. The concept of this path selection is illustrated in Fig. 5.

C. Retiming to Facilitate Wave-Pipelining Construction

With the false path checking method described above, we can check whether WP false paths can be formed from the selected single-period true paths on the left and right of flip-flop ff_i. If such WP false paths can be formed, we mark the original single-period clocking true paths forming them as *relevant paths*. Other singe-period clocking paths are called *irrelevant paths*. An example of the relevant and irrelevant paths is shown in Fig. 6(a). To construct WP with relevant paths, the flip-flop ff_i in the middle can be removed. Unfortunately, the removal of ff_i makes all the paths connected with it WP. Since many short paths may exist on the left and on the right of ff_i, connecting them by removing ff_i directly generates many paths whose delays are too small to meet the lower bound of the WP constraint (1). In addition, the removal of the flip-flop leads to a clustering of WP paths, which are vulnerable to be identified by attackers. Fig. 6(a) illustrates a construction example, where a WP false path is formed from the relevant paths after ff_i is removed. However, the irrelevant paths on the left and the right of ff_i lead to unnecessary WP generation. To deal with the challenges, we try to maintain the irrelevant paths on the left and right of ff_i to be single-period as much as possible.

To maintain the irrelevant paths as single-period clocking paths, we apply the retiming technique [54] to block these paths with flip-flops. Retiming transforms the structure of a circuit by moving the locations of flip-flops while preserving the function of this circuit. This concept is illustrated in Fig. 6(b), where the flip-flop ff_i is moved to the left of the OR gate. To maintain the original function of the circuit, a flip-flop is inserted at each input of the OR gate.

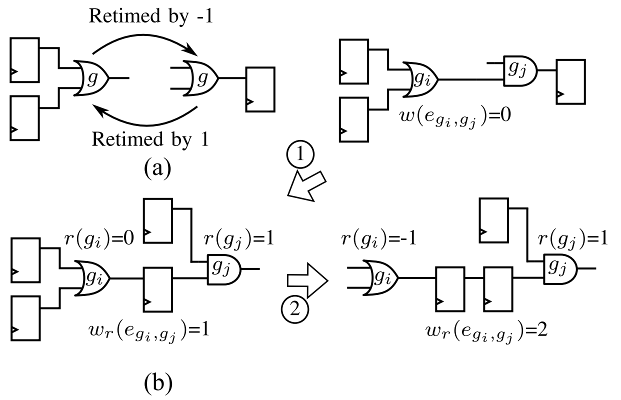


Fig. 7. Operations of retiming. (a) Basic operations of retiming. (b) Two retiming cases.

The concept of retiming can be explained using Fig. 7. To apply this technique, we use $g \in G$ to represent a combinational gate and a net $e \in E$ to represent the net connecting the output of a combinational gate and an input of another combinational gate. The delays from the input pins of a gate to its output pin are different, due to the internal structure of the gate. Therefore, these delays should be set individually according to the corresponding lookup table. Since the input pins of a gate appear along different combinational paths, we use d_g^p to represent the pin-to-pin delay for gate g along the path p. Interconnect delays can be modeled as extra nodes between nets, similar to combinational gates. Each net e_{g_i,g_j} between gates g_i and g_j has a weight $w(e_{g_i,g_j})$ to represent the number of flip-flops along the connection. Assume that between two gates g_i and g_j, there is a path p. The propagation delay of this path is equal to the sum of the delays of the gates on the path, expressed as follows:

$$d(p) = \sum_{k=1}^{n} d_{g_k}^{p} \tag{4}$$

where n is the number of gates on the path. Furthermore, the weight of a path p, representing the total number of flip-flops on the path, is defined as the sum of the weights of the nets along the path, expressed as

$$w(p) = \sum_{k=1}^{n-1} w(e_k). \tag{5}$$

The goal of retiming is to find an assignment of an integer $r(g)$ for each gate g to transform a circuit to another functionally equivalent circuit. $r(g)$ defines how many flip-flops are moved from the output of a gate to its inputs. In Fig. 7(a), the combinational gate is retimed by -1 if the flip-flops at its inputs are moved to its output. In this case, the integer $r(g)$ of the gate is equal to -1. On the contrary, the integer $r(g)$ of the gate is 1, if the flip-flop at its output is moved to its inputs.

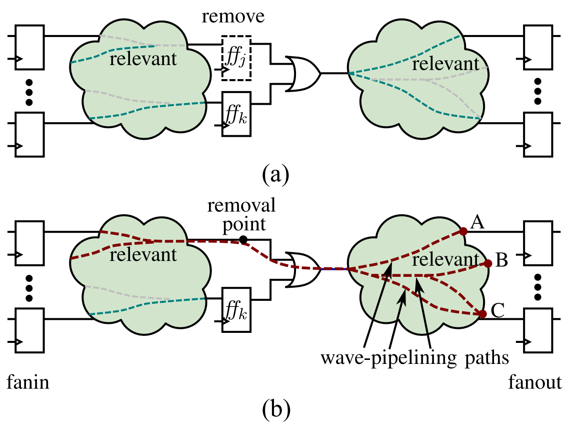
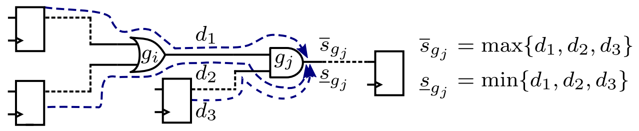


Fig. 8. Removal of flip-flops together with retiming. (a) ff_j is removed for WP construction. (b) WP paths can be constructed from the relevant paths and a small number of irrelevant paths after the retimed flip-flop ff_j is removed.



g_i
d_1
d_2
d_3
g_j
$\overline{s}_{g_j}$
$\underline{s}_{g_j}$
$\overline{s}_{g_j} = \max\{d_1, d_2, d_3\}$
$\underline{s}_{g_j} = \min\{d_1, d_2, d_3\}$

Fig. 9. Latest and the earliest arrival times of g_j.

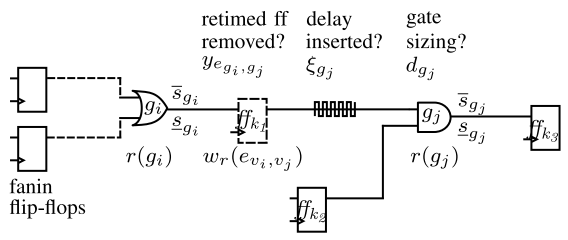


Fig. 10. Removal of a retimed flip-flop.

After retiming, the number of flip-flops on a net between gates g_i and g_j is written as $w_r(e_{g_i,g_j}) = w(e_{g_i,g_j}) + r(g_j) - r(g_i)$. Two retiming cases are shown in Fig. 7(b). In the first case ①, the flip-flop at the output of g_j is moved to its inputs, so that the number of flip-flops between g_i and g_j can be derived as $w_r(e_{g_i,g_j}) = w(e_{g_i,g_j}) + r(g_j) - r(g_i) = 0 + 1 - 0 = 1$. In the second case ②, the flip-flops at the inputs of g_i are moved to its output further, so that the number of flip-flops between g_i and g_j is increased to $w_r(e_{g_i,g_j}) = w(e_{g_i,g_j}) + r(g_j) - r(g_i) = 0 + 1 - (-1) = 2$. For a legal retiming, the retimed weight $w_r(e_{g_i,g_j})$ must be non-negative. To meet a given clock period T for a retimed circuit, any path p with a delay larger than the given clock period T should have a retimed weight $w_r(p)$ larger than 0 to guarantee there is a flip-flop on it, so that this path does not affect the clock period.

To construct WP paths, retiming can be applied to move the locations of flip-flops to block irrelevant paths, so that these paths can still be maintained as single-period to avoid unnecessary WP generation. With the retimed flip-flops, the irrelevant paths that do not contribute to the construction of WP paths still work within one clock period. The relevant paths are used to construct WP paths by removing the retimed flip-flops. The details of this construction are explained in the following sections.

D. Wave-Pipelining Construction With Removal of Flip-Flops Combined With Retiming

Retiming can facilitate the WP construction by maintaining irrelevant paths as single-period as described above. We then construct WP paths by removing flip-flops combined with retiming. A simple construction example is illustrated in Fig. 8(a), where the retimed flip-flop ff_j is removed for WP and the retimed flip-flop ff_k is kept to block irrelevant paths. However, the removal of the retimed flip-flop ff_j generates many WP paths. To enlarge the delays of these paths to meet the lower bound defined in (1), a large area overhead might be incurred. Therefore, it is not straightforward to determine the optimal location to apply retiming to construct WP. To deal with this challenge, we formulate the WP construction as an ILP problem. The goal of the formulation is to achieve the construction of WP paths whose delays meet the timing constraints (1), (2) as well as the gray region requirement (3) with a minimum area overhead.

The removal of a retimed flip-flop leads to many WP paths from left and the right of the removal point, as shown in Fig. 8(b). To guarantee the timing constraints (1)–(3), all WP paths may be traversed to check their delays. However, this straightforward traversal might be time-consuming due to the number of paths. To accelerate this process, we assign two variables, $\overline{s}_g$ and $\underline{s}_g$ to represent the latest and the earliest arrival times at the output of a combinational gate g, as shown in Fig. 9. The latest arrival time at the gate g_j is the maximum delay with which the data from a flip-flop travels through the longest path and arrives at the output of this gate, denoted as $\overline{s}_{g_j} = \max\{d_1, d_2, d_3\}$. On the contrary, the earliest arrival time at the gate g_j is the minimum delay with which the data from a flip-flop travels through the shortest path and arrives at the output of this gate, denoted as $\underline{s}_{g_j} = \min\{d_1, d_2, d_3\}$. To tolerate PVT variations, for the WP paths after a flip-flop is removed, shown in Fig. 8(b), if the latest and the earliest arrival times at points A, B, and C satisfy the timing constraints (1), (2) and the gray region requirement (3), all the other WP paths resulting from the removal of ff_j are also guaranteed, because the arrival times of other paths are bounded between them.

Since we do not know at which location the retiming should be performed to construct WP paths, a variable $y_{e_{g_i,g_j}}$ is assigned for each net e_{g_i,g_j} to indicate whether the retimed flip-flop on this net can be removed, with $y_{e_{g_i,g_j}} = 1$ to indicate that the retimed flip-flop is removed and vice versa. If there is no retimed flip-flop on e_{g_i,g_j}, $y_{e_{g_i,g_j}}$ should be set to 0. Consequently, the relation between $y_{e_{g_i,g_j}}$ and the number of retimed flip-flops $w_r(e_{g_i,g_j})$ on this net can be established as follows:

$$y_{e_{g_i,g_j}} \le w_r(e_{g_i,g_j}). \quad (6)$$

With this setting, three representative cases for a net between gates g_i and g_j should be examined, as shown in Fig. 10. Detailed timing constraints of each case can be found in [62].

Case 1: The net from gate g_i to gate g_j has the retimed weight $w_r(e_{g_i,g_j}) = w(e_{g_i,g_j}) + r(g_j) - r(g_i) = 1$, and thus a retimed flip-flop ff_{k_1} exists along this net. The retimed flip-flop is not removed, denoted as $y_{e_{g_i,g_j}} = 0$.

Case 2: The net from gate g_i to g_j has the retimed weight $w_r(e_{g_i,g_j}) = w(e_{g_i,g_j}) + r(g_j) - r(g_i) = 1$, but the flip-flop is removed, denoted as $y_{e_{g_i,g_j}} = 1$.

Case 3: The net from gate g_i to g_j does not have a retimed flip-flop, $w_r(e_{g_i,g_j}) = 0$. In this case, the data at the output of g_i passes through g_j directly.

When removing flip-flops combined with retiming, each of the cases above can happen. We let the solver determine which case actually happens during WP construction. After this construction, no flip-flop should appear on the relevant paths. Consequently, if there is a retimed flip-flop on a net along a relevant path $w_r(e_{g_i,g_j}) = 1$, this flip-flop should be removed, so that $y_{e_{g_i,g_j}} = 1$. Accordingly, the following constraints should be met:

$$w_r\left(e_{g_i,g_j}\right) = y_{e_{g_i,g_j}} \quad \forall e_{g_i,g_j} \in E \text{ on relevant paths.} \tag{7}$$

We thus formulate the WP construction problem as follows:

$$\text{Minimize } \alpha \sum_{g \in G} \xi_g - \beta \sum_{g \in G} \sum_{i=1}^{n_g} d_g^i + \gamma \sum_{g \in G} r(g)(\#\text{input of } g - \#\text{output of } g). \tag{8}$$

$$\text{Subject to (6) and gray region constraints} \tag{9}$$

$$\text{Case 1 constraints, if } w_r\left(e_{g_i,g_j}\right) + y_{e_{g_i,g_j}} = 1 \tag{10}$$

$$\text{Case 2 constraints, if } w_r\left(e_{g_i,g_j}\right) + y_{e_{g_i,g_j}} = 2 \tag{11}$$

$$\text{Case 3 constraints, if } w_r\left(e_{g_i,g_j}\right) + y_{e_{g_i,g_j}} = 0 \tag{12}$$

where ξ_g is introduced to enlarge the delay of WP paths, which can be implemented by lengthening interconnects. d_g^i is the delay from the ith input pin to the output pin of gate g. n_g is the number of input pins for g. $r(g)(\#\text{input of } g - \#\text{output } of\ g)$ represents the increased number of retimed flip-flops. α, β, and γ are constants with $\alpha \geq \gamma \geq \beta$ to prevent the exposure of WP resulting from lengthened interconnects and suppress area overhead by more retimed flip-flops and gate sizing. The conditional constraints (10)–(12) can be converted into linear constraints as described in [55].

Since only one flip-flop is used to construct WP at a time and other flip-flops are kept in the circuit as shown in Fig. 8(b), the part of a circuit around this flip-flop for WP construction is not large. Therefore, we solve (8)–(10) directly with an ILP solver to construct WP.

E. Wave-Pipelining Construction With Duplication Combined With Retiming

The WP construction by applying the removal of flip-flops combined with retiming described above might not be achieved successfully for a flip-flop ff_i, due to the circuit structure and the restriction on the area overhead incurred by lengthening interconnects, so that the ILP formulation (8)–(10) may return no solution. To solve this problem, retiming is first applied to block irrelevant paths as much as possible and the circuit is duplicated in part to bypass such paths further to facilitate WP construction. To implement the first step, the variable y_e, which indicates whether the retimed flip-flop on a net is removed, should be set to 0 for all nets in a circuit to guarantee that all paths are still single-period. The original flip-flop ff_i should be moved to its left side to block irrelevant paths as much as possible with the formulation (8)–(10), as shown in Fig. 11(a). This leftward movement of flip-flops leads to fewer logic gates to be duplicated, as shown in Fig. 11(b).

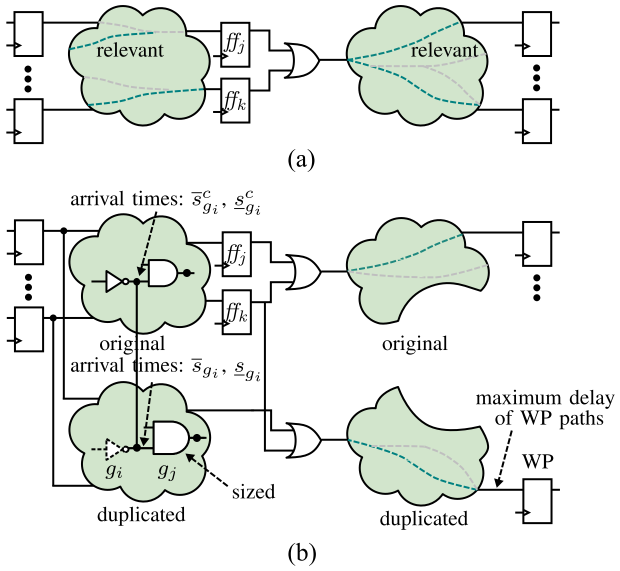

relevant
ff_j
ff_k
relevant
(a)
arrival times: $\overline{s}_{g_i}^c$, $\underline{s}_{g_i}^c$
ff_j
ff_k
original
original
arrival times: $\overline{s}_{g_i}$, $\underline{s}_{g_i}$
maximum delay of WP paths
WP
g_i
g_j
sized
duplicated
duplicated
(b)

Fig. 11. WP construction with logic duplication and gate sizing combined with retiming. (a) ff_i is moved to the left of the OR gate after retiming. (b) Logic duplication and gate sizing.

In the second step, after retiming, we duplicate the logic in the circuit and size the gates and lengthen interconnects so that the delays of all WP paths meet timing constraints (1)–(3) as illustrated in Fig. 11(b). In the duplicated circuit on the right of retimed flip-flops, we only keep the flip-flops at which WP paths terminate. The other flip-flops stay in the original circuit. Afterward, we delete the logic gates backwards to remove those gates that do not drive any flip-flop to reduce resource usage. When duplicating the logic on the left of ff_j, however, we need to keep all the logic gates to maintain the correct function of the circuit.

In the duplicated logic in Fig. 11(b), we do not duplicate flip-flops. Therefore, all combinational paths in the duplicated logic are WP paths and their delays should meet the timing constraints (1)–(3). To meet these constraints, we size the gates and lengthen interconnects in the duplicated logic with an ILP formulation. For example, in Fig. 11(b), we assume the latest and the earliest arrival times at the output of the inverter in the duplicated circuit as $\overline{s}_{g_i}$ and $\underline{s}_{g_i}$. Similarly, we assume the latest and the earliest arrival times at the output of the AND gate in the duplicated circuit as $\overline{s}_{g_j}$ and $\underline{s}_{g_j}$. Furthermore, the delay from an input pin to the output pin of the AND gate along the path p traveling through the inverter is written as $d_{g_j}^p$. With these definitions, the arrival times between the inverter

TABLE I
RESULTS OF CONSTRUCTING WP PATHS

Circuit			WP True Construction			WP False Construction			Cost			Runtime
	n_s	n_g	n_{wpt}	n_t	n'_t	n_{wpf}	n_f	n'_f	n_p	n_d	n_r	$t_r(s)$
s35932	1728	16065	579	80213	80792	715	130087	130802	23.4	35	2	481.8
s38584	1452	19253	420	202647	203067	2486	722378	724864	9.7	63	4	736.2
s38417	1636	22179	8369	637091	645460	9837	594078	603915	18.9	105	0	643.8
s15850	534	9772	1932	1693510	1695442	606	20999926	21000532	98.6	0	4	699.0
s13207	669	7951	7830	294531	302361	13410	728262	741672	12.5	74	4	223.4
s9234	228	5597	282	5710	5992	1349	154825	156174	27.2	62	2	362.3
s5378	179	2779	844	8465	9309	706	467	1173	49.3	73	0	114.9
s4863	104	2342	300	359663	359963	0	70432413	70432413	38.9	26	5	3564.7
s1423	74	657	278	7997	8275	272	5698	5970	45.7	91	2	19.3
s1238	18	508	4	381	385	2	849	851	19.9	14	5	0.8

and the AND gate can be written as

$$\overline{s}_{g_j} \geq \overline{s}_{g_i} + \xi_{g_j} + d^p_{g_j} \tag{13}$$

$$\underline{s}_{g_j} \leq \underline{s}_{g_i} + \xi_{g_j} + d^p_{g_j}. \tag{14}$$

To reduce the number of duplicated gates, we try to connect the output pins of logic gates in the duplicated logic to the original gates as much as possible, as illustrated in Fig. 11(b). In the original logic, the latest and the earliest arrival times are constants. Assume that the arrival times at the output of the inverter in the original circuit are $\overline{s}^c_{g_i}$ and $\underline{s}^c_{g_i}$, and a 0-1 variable p_i indicates whether the output pin in the duplicated logic should be driven by the original logic. We can then extend constraints (13) and (14) as

$$\overline{s}_{g_j} \geq \overline{s}_{g_i} + d^p_{g_j} + \xi_{g_j} - p_i M \tag{15}$$

$$\overline{s}_{g_j} \geq \overline{s}^c_{g_i} + d^p_{g_j} + \xi_{g_j} - (1 - p_i) M \tag{16}$$

$$\underline{s}_{g_j} \leq \underline{s}_{g_i} + d^p_{g_j} + \xi_{g_j} + p_i M \tag{17}$$

$$\underline{s}_{g_j} \leq \underline{s}^c_{g_i} + d^p_{g_j} + \xi_{g_j} + (1 - p_i) M \tag{18}$$

where M is a very large positive constant used to transform the conditional constraints to linear constraints [56]. In either case when an output pin in the duplicated logic is connected or disconnected with the original logic, only two constraints in (15)–(18) are valid.

In the description above, we allow a path delay to be extended with lengthening interconnects. However, we try to keep the delay incurred by interconnects as small as possible. In addition, we try to reduce the overall area overhead when implementing WP. Therefore, we formulate the construction problem as

$$\text{minimize} \;\; \alpha \sum_{g \in G} \xi_g - \beta \sum_{g \in G} \sum_{i=1}^{n_g} d^i_g - \gamma \sum_{i \in I} p_i \tag{19}$$

$$\text{subject to} \;\; (37)\text{–}(38)\ ,\ (40) \text{ and } (15)\text{–}(18) \tag{20}$$

where $\alpha \geq \gamma \geq \beta$ to prevent the exposure of WP resulting from lengthened interconnects and duplicated gates and suppress area overhead by gate sizing. In this setting, the effectiveness of camouflaging is more important than incurred area overhead. I is the index set of all output pins. After the ILP problem above is solved, the gates that do not drive any other gates in the duplicated logic are removed to reduce resource usage. With this extra step, WP can also be constructed even if the optimization problem in (8)–(10) returns no solution.

In the ILP formulation (8)–(12) and (19)-(20), we assume constant input slews and output loads for gate delays for simplification. However, we do not know which combination of input slews and output loads should be adopted for the delay reference of a gate in the ILP formulation. For simplification, the input slew and output load values in the middle of their corresponding ranges in the lookup table are used as typical values to select the delay for WP construction. After the WP paths are determined, we verify the timing constraints of these paths using real lookup tables indexed by slew and load values. For each input pin to the output pin of a gate, the delay and output slew is set according to real input slew and output load using the corresponding lookup tables. Due to the assumption of constant delays in the ILP formulation, the real delays of WP paths may not meet the timing constraints. In the case of timing violation, we first size the logic gates iteratively to modify the path delay. If this is still not sufficient to solve the timing violation, buffers are then added or removed to meet WP constraints. If the timing constraints are still not met with gate sizing and buffer insertion described above, we switch to another flip-flop to construct WP paths. However, the construction process is not guaranteed to converge.

VI. Experimental Results

TimingCamouflage+ was implemented in C++ and tested using a 3.20-GHz CPU. We demonstrate the results using circuits from the ISCAS89 benchmark set. The number of flip-flops and the number of logic gates are shown in the columns $\boldsymbol{n_s}$ and $\boldsymbol{n_g}$ in Table I, respectively. The benchmark circuits were sized using a 45 nm library. We set the timing margin δ to 0.15 to tolerate PVT variations and the inaccuracy factor τ of delay estimation of attackers to 0.2. To simplify the delay models, input slews and output loads are set to constant values. However, TimingCamouflage+ is independent of delay characterization and can work with any delay model. We used Gurobi [55] to solve the optimization problems.

The results of WP path construction are shown in Table I. The column $\boldsymbol{n_{wpt}}$ shows the numbers of WP true paths whose delays are in the gray region. These paths are used to guarantee that attackers must perform testing to distinguish WP true paths from single-period clocking true paths. Without these paths, attackers can assume all testable paths are single-period and avoid the expensive test procedure. The column $\boldsymbol{n_t}$ shows the numbers of single-period clocking true paths whose delays meet the gray region requirement (3). When attackers try to detect the locations of WP paths, these true paths need to be tested to determine whether their delays are actually larger or smaller than T. The column $\boldsymbol{n'_t} = \boldsymbol{n_{wpt}} + \boldsymbol{n_t}$, is the total number of suspicious true paths that are required to be tested.

The column $\boldsymbol{n_{wpf}}$ shows the numbers of WP false paths whose delays are in the gray region. In the experiments, we set the target numbers of WP true and false paths both to 100 and the threshold distance dis_t between the flip-flops in the first line of Fig. 4 to construct WP true and false paths to ten times of the minimum distance between all pairs of flip-flops. We executed the construction of WP true and false paths shown in Fig. 4 repeatedly using the method described in Sections V-D and V-E. When we constructed WP false paths, we also found WP true paths in the circuit and vice versa. Consequently, the numbers of these paths shown in the columns $\boldsymbol{n_{wpt}}$ and $\boldsymbol{n_{wpf}}$ are larger than 100 for many test cases except s4863 and s1238. In s4863 there is no WP false path and in s1238 the number of WP paths is very small due to the limited sizes of these two circuits. In all large test cases, however, WP paths have been constructed successfully. In practice, as the circuit size increases, more path candidates become available for the WP false path construction, so that WP false paths can always be constructed successfully.

The column $\boldsymbol{n_f}$ shows the number of suspicious single-period clocking false paths. Since their delays meet the gray region requirement (3), these paths are suspicious WP false paths for attackers. The total suspicious WP false paths are shown in $\boldsymbol{n'_f} = \boldsymbol{n_{wpf}} + \boldsymbol{n_f}$. To determine whether a false path is WP or not from a huge number of suspicious paths, much effort is required as explained in Section IV, since these false paths cannot be triggered for delay test.

The column $\boldsymbol{n_p}$ shows the equivalent number of inserted delays in the unit of the delay of a buffer by lengthening interconnects to enlarge WP path delays. Since the number of inserted delays does not increase with respect to circuit size, the area cost for constructing WP paths is negligible in relatively large circuits.

In the experiments, we constrained the inserted delay ξ_g incurred by lengthening interconnects to be no larger than the delay of three buffers for each net. With this constraint, the removal of flip-flops combined with retiming in Section V-D can only be achieved in s15850 to construct WP paths. In the remaining circuits, this method cannot construct WP successfully, since a few short paths still require much delay insertion to meet the timing constraint of WP. However, the number of such paths is very small, confirmed by the total delays inserted as shown in column $\boldsymbol{n_p}$. In this case, circuit duplication combined with retiming is applied as described in Section V-E. The column $\boldsymbol{n_d}$ in Table I shows the number of logic gates duplicated to construct WP. Since we only inserted WP paths at a limited number of locations, generally the number of duplicated gates does not increase with respect to circuit size.

To construct WP paths, retiming might cause an increase of the number of flip-flops, shown in column $\boldsymbol{n_r}$. This increase and the corresponding area overhead are still negligible. The last column $\boldsymbol{t_r}$ in Table I shows the runtime of TimingCamouflage+, which is acceptable because WP construction is a one-time effort.

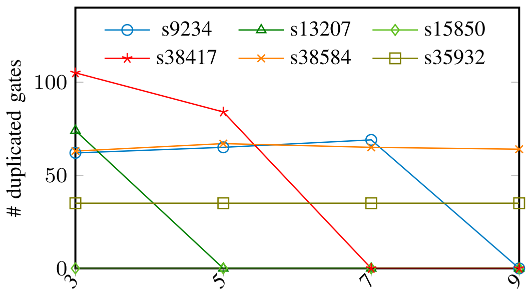


Fig. 12. Number of duplicated gates over the increase of interconnect delays (in unit of buffer delays).

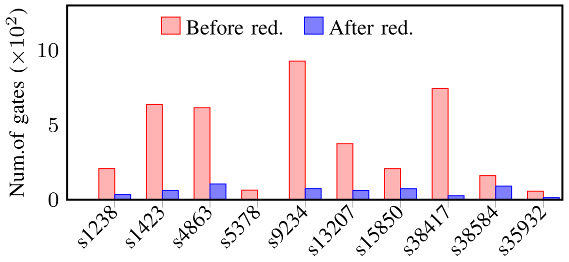


Fig. 13. Comparison of gate numbers before/after reduction.

We also applied TimingCamouflage+ in two practical circuits, vga_lcd and pci_bridge32, from TAU 2013 variation-aware timing analysis contest. In both circuits, WP false and true paths can be constructed successfully. Specifically, in vga_lcd, 169 WP true paths and 138 WP false paths were constructed with 25 inserted buffers, 60 duplicated gates and 5 more retimed flip-flops. In pci_bridge32, 206 WP true paths and 1162 WP false paths were constructed with 18 inserted buffers, 43 duplicated gates and 2 more retimed flip-flops.

When constructing WP paths by lengthening interconnects, we constrained the interconnect delay ξ_g inserted before a gate g. Fig. 12 shows the changes of the numbers of duplicated gates when the interconnect delays are relaxed. It is clear that with the increase of delays inserted before a gate g, the number of duplicated gates required to construct WP paths is decreased in three circuits. When the allowable number of delay units reaches 9, no circuit duplication is required for all these circuits. In s15850, duplicated gates are not even required to construct the WP paths when the allowable number of delay units is larger than or equal to 3, due to the efficiency of the removal of flip-flops combined with retiming in Section V-D. For s38584 and s35932, the removal of flip-flops combined with retiming cannot be achieved successfully due to circuit structures. Therefore, the numbers of duplicated gates are not reduced.

In the WP construction formulation (19)–(20), we maximize the number of output pins that can be driven by the original circuit as illustrated in Fig. 11(b). Consequently, the number of logic gates in the duplicated circuit can be reduced. Fig. 13 compares the numbers of gates in the originally duplicated circuit before the removed flip-flop in Fig. 11 and the number of gates after reduction by using the original circuit. In all test cases, the numbers of duplicated gates were reduced significantly.

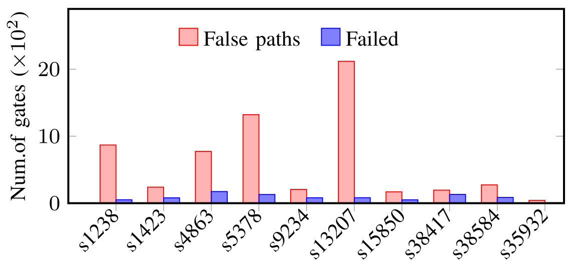


Fig. 14. Results of false path sizing attack.

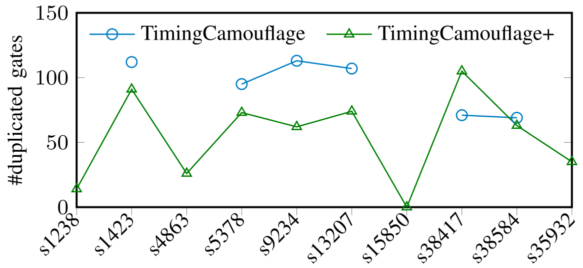


Fig. 15. Comparison of the number of duplicated gates with TimingCamouflage in [1].

In the experiments, we also simulated the gate sizing attack on the netlist as discussed in Section IV. The basic idea was that all false paths with delays falling in the gray region were treated as WP paths and their delays were sized to meet (1)–(2). The results of this simulated attack are shown in Fig. 14, where the first bar shows the number of false paths we used to simulate the attack. The second bar shows the number of false paths that were not sized successfully. Even with such a small number of false paths from the huge set of false paths in the original circuit, no sizing attack succeeded.

TimingCamouflage in [1] uses the same timing concept to camouflage the netlists. However, it constructs WP paths with duplicated gates without applying the retiming technique. Consequently, it requires many duplicated gates and interconnect delays to construct the WP paths. To verify the improvement of incorporating retiming in TimingCamouflage+, TimingCamouflage in [1] was implemented with the maximum allowable inserted delay units constrained the same as in TimingCamouflage+. The results are shown in Figs. 15 and 16. Since the maximum numbers of inserted delay units were constrained to be no larger than three, TimingCamouflage in [1] fails to construct the WP paths in s1238, s4863, s15850, and s35932, as shown in Fig. 15. In the other cases, the numbers of duplicated gates with TimingCamouflage+ are also smaller than those with TimingCamouflage in [1], except for s38417, in which the number of duplicated gates is slightly larger in TimingCamouflage+. The comparison of inserted delay units are shown in Fig. 16, where the numbers with TimingCamouflage+ are consistently smaller in all test cases than with [1]. The comparison of area overhead to construct WP paths are shown in Fig. 17, where the area overhead with TimingCamouflage+ is smaller than that with TimingCamouflage in [1], except for s38417. For s1238, s4863, s15850, and s35932, WP paths cannot be constructed

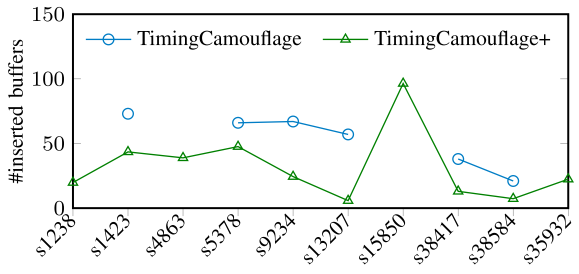


Fig. 16. Comparison of the number of inserted delay units with TimingCamouflage in [1].

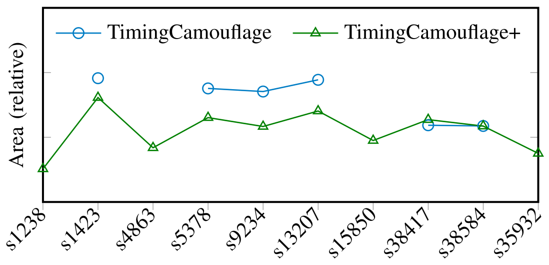


Fig. 17. Comparison of area overhead with TimingCamouflage in [1].

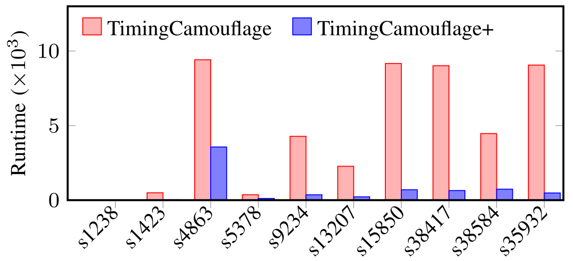


Fig. 18. Comparison of runtime with TimingCamouflage in [1].

successfully with TimingCamouflage with the maximum numbers of inserted delay units to be no larger than three. The power consumption is roughly proportional to the area overhead.

In TimingCamouflage+, the construction of WP paths is accelerated by sorting and filtering flip-flops described in Section V-A. Consequently, the runtime is reduced significantly. The comparison of computation time with TimingCamouflage [1] is shown in Fig. 18. From this comparison, it is clear that TimingCamouflage+ outperforms TimingCamouflage in [1] in efficiency.

VII. Conclusion

In this article, we have proposed TimingCamouflage+ to secure circuit netlists against counterfeiting. Since a netlist itself does not carry all design information anymore, the difficulty of attack has increased significantly due to additional test cost and the introduced WP false paths. TimingCamouflage+ opens up a new dimension of netlist camouflage at the circuit level, and it is fully compatible with other previous counterfeiting methods so that they can be combined together to strengthen netlist security. Future work includes experimental attempts of the attacks discussed in Section IV and the combination of the proposed camouflage method with other existing methods. Advanced timing concepts can be embedded into the netlist to enhance circuit security [57]–[61].

References

[1] G. L. Zhang, B. Li, B. Yu, D. Z. Pan, and U. Schlichtmann, "TimingCamouflage: Improving circuit security against counterfeiting by unconventional timing," in *Proc. Design Autom. Test Europe Conf.*, 2018, pp. 91–96.

[2] A. Vijayakumar, V. C. Patil, D. E. Holcomb, C. Paar, and S. Kundu, "Physical design obfuscation of hardware: A comprehensive investigation of device and logic-level techniques," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 12, no. 1, pp. 64–77, Jan. 2017.

[3] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson, "Stealthy dopant-level hardware trojans: Extended version," *J. Cryptograph. Eng.*, vol. 4, no. 1, pp. 19–31, 2014.

[4] L.-W. Chow, J. P. Baukus, J. William, and M. Clark, "Integrated circuits protected against reverse engineering and method for fabricating the same using an apparent metal contact line terminating on field oxide," U.S. Patent 7 294 935, 2007.

[5] S. Malik, G. T. Becker, C. Paar, and W. P. Burleson, "Development of a layout-level hardware obfuscation tool," in *Proc. Comput. Soc. Annu. Symp. VLSI*, 2015, pp. 204–209.

[6] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Proc. Conf. Comput. Commun. Security*, 2013, pp. 709–720.

[7] J. Rajendran, O. Sinanoglu, and R. Karri, "VLSI testing based security metric for IC camouflaging," in *Proc. Int. Test Conf.*, 2013, pp. 1–4.

[8] Y. Lee and N. A. Touba, "Improving logic obfuscation via logic cone analysis," in *Proc. Latin–Amer. Test Symp.*, 2015, pp. 1–6.

[9] M. Li *et al.*, "Provably secure camouflaging strategy for IC protection," in *Proc. Int. Conf. Comput.-Aided Des.*, 2016, pp. 28–35.

[10] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "CamoPerturb: Secure IC camouflaging for minterm protection," in *Proc. Int. Conf. Comput.-Aided Des.*, 2016, pp. 1–8.

[11] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "Cyclic obfuscation for creating sat-unresolvable circuits," in *Proc. Great Lakes Symp. VLSI*, 2017, pp. 173–178.

[12] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately deobfuscating integrated circuits," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust (HOST)*, 2017, pp. 95–100.

[13] Y. Shen and H. Zhou, "Double DIP: Re-evaluating security of logic encryption algorithms," in *Proc. Great Lakes Symp. VLSI*, 2017, pp. 179–184.

[14] A. Chakraborty, Y. Liu, and A. Srivastava, "TimingSAT: Timing profile embedded SAT attack," in *Proc. Int. Conf. Comput.-Aided Design*, 2018, pp. 1–6.

[15] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proc. Design Autom. Conf.*, 2012, pp. 83–89.

[16] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending piracy of integrated circuits," in *Proc. Design Autom. Test Europe Conf.*, 2008, pp. 1069–1074.

[17] S. Dupuis, P.-S. Ba, G. D. Natale, M.-L. Flottes, and B. Rouzeyre, "A novel hardware logic encryption technique for thwarting illegal overproduction and hardware trojans," in *Proc. Int. On-Line Test. Symp.*, 2014, pp. 49–54.

[18] S. M. Plaza and I. L. Markov, "Solving the third-shift problem in IC piracy with test-aware logic locking," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 6, pp. 961–971, Jun. 2015.

[19] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC piracy using reconfigurable logic barriers," *IEEE Design Test Comput.*, vol. 27, no. 1, pp. 66–75, Jan./Feb. 2010.

[20] Y. Xie and A. Srivastava, "Delay locking: Security enhancement of logic locking against ic counterfeiting and overproduction," in *Proc. Design Autom. Conf.*, 2017, pp. 1–6.

[21] R. S. Chakraborty and S. Bhunia, "HARPOON: An obfuscation-based SoC design methodology for hardware protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 28, no. 10, pp. 1493–1502, Oct. 2009.

[22] M. Zaman, A. Sengupta, D. Liu, O. Sinanoglu, Y. Makris, and J. J. V. Rajendran, "Towards provably-secure performance locking," in *Proc. Design Autom. Test Europe Conf.*, 2018, pp. 1558–1101.

[23] A. Chakraborty, Y. Xie, and A. Srivastava, "GPU obfuscation: Attack and defense strategies," in *Proc. Design Autom. Conf.*, 2018, pp. 1–6.

[24] N. G. Jayasankaran, A. S. Borbon, A. S. Borbon, J. Hu, and J. Rajendran, "Towards provably-secure analog and mixed-signal locking against overproduction," in *Proc. Int. Conf. Comput.-Aided Design*, 2018, pp. 1–8.

[25] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust (HOST)*, 2015, pp. 137–143.

[26] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. J. V. Rajendran, "Security analysis of Anti-SAT," in *Proc. Asia South Pac. Design Autom. Conf.*, 2017, pp. 342–347.

[27] X. Xu, B. Shakya, M. M. Tehranipoor, and D. Forte, "Novel bypass attack and BDD-based tradeoff analysis against all known logic locking attacks," in *Proc. Cryptograph. Hardw. Embedded Syst.*, 2017, pp. 189–210.

[28] D. Kirovski and M. Potkonjak, "Local watermarks: Methodology and application to behavioral synthesis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 9, pp. 1277–1283, Sep. 2003.

[29] F. Koushanfar and G. Qu, "Hardware metering," in *Proc. Design Autom. Conf.*, 2001, pp. 1–6.

[30] X. Wang, Y. Guo, T. Ramhan, D. Zhang, and M. Tehranipoor, "DOST: Dynamically obfuscated wrapper for split test against IC piracy," in *Proc. IEEE Asian Hardw. Orient. Security Trust Symp. (AsianHOST)*, 2017, pp. 1–6.

[31] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *Proc. Design Autom. Test Europe Conf.*, 2013, pp. 1259–1264.

[32] H.-Y. Hsieh, W. Liu, R. K. Cavin, III, and C. T. Gray, "Concurrent timing optimization of latch-based digital systems," in *Proc. Int. Conf. Comput. Des.*, 1995, pp. 680–685.

[33] W. P. Burleson, M. Ciesielski, F. Klass, and W. Liu, "Wave-pipelining: A tutorial and research survey," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 6, no. 3, pp. 464–474, Sep. 1998.

[34] G. Seetharaman and B. Venkataramani, "Automation schemes for FPGA implementation of wave-pipelined circuits," *ACM Trans. Reconfig. Technol. Syst.*, vol. 2, no. 2, pp. 1–19, 2009.

[35] G. L. Zhang, B. Li, M. Hashimoto, and U. Schlichtmann, "VirtualSync: Timing optimization by sychronizing logic waves with sequential and combinational components as delay units," in *Proc. Design Autom. Conf.*, 2018, pp. 1–6.

[36] F. Yuan and Q. Xu, "On timing-independent false path identification," in *Proc. Int. Conf. Comput.-Aided Des.*, 2010, pp. 532–535.

[37] D. H. Du, S. H. Yen, and S. Ghanta, "On the general false path problem in timing analysis," in *Proc. Design Autom. Conf.*, 1989, pp. 555–560.

[38] K. Heragu, J. H. Patel, and V. D. Agrawal, "Fast identification of untestable delay faults using implications," in *Proc. Int. Conf. Comput.-Aided Des.*, 1997, pp. 642–647.

[39] X. Zhang. (2003). *High Performance Low Leakage Design Using Power Compiler and Multi-Vt Libraries*. [Online]. Available: http://www.synopsys.com/

[40] R. Puri, "Minimizing power under performance constraint," in *Proc. Int. Conf. Integr. Circuit Design Technol.*, 2004, pp. 159–163.

[41] F. G. F. Sill and D. Timmermann, "Reducing leakage with mixed-VTH (MVT)," in *Proc. Int. Conf. VLSI Design*, 2005, pp. 874–877.

[42] I. R. Nirmala, D. Vontela, S. Ghosh, and A. Iyengar, "A novel threshold voltage defined switch for circuit camouflaging," in *Proc. IEEE Eur. Test Symp. (ETS)*, 2016, pp. 1558–1780.

[43] M. Li, K. Shamsi, Y. Jin, and D. Z. Pan, "TimingSAT: Decamouflaging timing-based logic obfuscation," in *Proc. Int. Test Conf.*, 2018, pp. 1–10.

[44] M. D. Silva, M.-L. Flottes, G. D. Natale, and B. Rouzeyre, "Preventing scan attacks on secure circuits through scan chain encryption," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 3, pp. 538–550, Mar. 2019.

[45] B. Yang, K. Wu, and R. Karri, "Secure scan: A design-for-test architecture for crypto chips," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 10, pp. 2287–2293, Oct. 2006.

[46] M. E. Massad, M. E. Massad, and M. E. Massad, "Reverse engineering camouflaged sequential circuits without scan access," in *Proc. Int. Conf. Comput.-Aided Des.*, 2017, pp. 22–40.

[47] K. Shamsi, M. Li, D. Z. Pan, and Y. Jin, "KC2: Key-condition crunching for fast sequential circuit deobfuscation," in *Proc. Design Autom. Test Europe Conf.*, 2019, pp. 534–539.

[48] Y. Xie and A. Srivastava, "Anti-SAT: Mitigating SAT attack on logic locking," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 2, pp. 199–207, Feb. 2019.

[49] Y. Xie and A. Srivastava, "Mitigating SAT attack on logic locking," in *Proc. Cryptograph. Hardw. Embedded Syst.*, 2016, pp. 127–146.

[50] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan, "SMT attack: Next generation attack on obfuscated circuits with capabilities and performance beyond the SAT attacks," in *Proc. Conf. Cryptograph. Hardw. Embedded Syst.*, 2019, pp. 97–122.

[51] S. Devadas, K. Keutzer, and S. Malik, "Computation of floating mode delay in combinational circuits: Theory and algorithms," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 12, no. 12, pp. 1913–1923, Dec. 1993.

[52] O. Coudert, "An efficient algorithm to verify generalized false paths," in *Proc. Design Autom. Conf.*, 2010, pp. 188–193.

[53] K. S. Kim, S. Mitra, and P. G. Ryan, "Delay defect characteristics and testing strategies," *IEEE Des. Test. Comput.*, vol. 20, no. 5, pp. 8–16, Sep./Oct. 2003.

[54] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *J. Algorithmica*, vol. 6, pp. 5–35, Jun. 1991.

[55] Gurobi Optimization, Inc. (2013). *Gurobi Optimizer Reference Manual*. [Online]. Available: http://www.gurobi.com

[56] D. Chen, R. Batson, and Y. Dang, *Applied Integer Programming: Modeling and Solution*. Hoboken, NJ, USA: Wiley, 2011.

[57] G. L. Zhang, B. Li, and U. Schlichtmann, "Sampling-based buffer insertion for post-silicon yield improvement under process variability," in *Proc. Design Autom. Test Europe Conf.*, 2016, pp. 1457–1460.

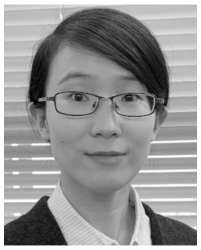
[58] B. Li and U. Schlichtmann, "Statistical timing analysis and criticality computation for circuits with post-silicon clock tuning elements," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 11, pp. 1784–1797, Nov. 2015.

[59] G. L. Zhang, B. Li, Y. Shi, J. Hu, and U. Schlichtmann, "EffiTest2: Efficient delay test and prediction for post-silicon clock skew configuration under process varaitions," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 4, pp. 705–718, Apr. 2019.

[60] N. Chen, B. Li, and U. Schlichtmann, "Iterative timing analysis based on nonlinear and interdependent flipflop modeling," *IET Circuits Devices Syst.*, vol. 6, no. 5, pp. 330–337, Sep. 2012.

[61] G. L. Zhang, B. Li, and U. Schlichtmann, "PieceTimer: A holistic timing analysis framework considering setup/hold time interdependency using a piecewise model," in *Proc. Int. Conf. Comput.-Aided Des.*, 2016, pp. 1–8.

[62] [Online]. Available: https://edawww.regent.e-technik.tu-muenchen.de/public/upload/202002231501_appendix_to_DOI10.1109TCAD.2020.2974338_arxiv.pdf



Grace Li Zhang received the Dr.-Ing. degree from the Technical University of Munich (TUM), Munich, Germany, in 2018.

She is currently a Postdoctoral Researcher with the Chair of Electronic Design Automation, TUM. Her current research interests include high-performance and low-power design, as well as emerging systems.

Bing Li received the Dr.-Ing. and Habilitation degrees from the Technical University of Munich (TUM), Munich, Germany, in 2010 and 2018, respectively.

He is currently a Researcher with the Chair of Electronic Design Automation, TUM. His current research interests include high-performance and low-power design of integrated circuits and systems.

Meng Li (Member, IEEE) received the Ph.D. degree from the University of Texas at Austin, Austin, TX, USA, in 2018, under the supervision of Dr. D. Z. Pan.

He is currently an AI Research Scientist with Facebook, Menlo Park, CA, USA. His research interests include AI software/hardware co-design, hardware-oriented security, and reliability.

Dr. Li was a recipient of the UT Austin Margarida Jacome Outstanding Dissertation Prize in 2019, the EDAA Outstanding Dissertation Award in 2019, the First Place in the Grand Final of ACM Student Research Competition in 2018, the Best Poster Award in ASPDAC Ph.D. forum in 2018, and the UT Austin University Graduate Fellowship in 2013, as well as the Best Paper Award in HOST'2017 and GLSVLSI'2018.

Bei Yu received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Assistant Professor with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Hong Kong. His research interests include machine learning and deep learning with applications in VLSI CAD.

David Z. Pan (Fellow, IEEE) received the B.S. degree from Peking University, Beijing, China, and the M.S. and Ph.D. degrees from the University of California at Los Angeles, Los Angeles, CA, USA.

He is currently an Engineering Foundation Professor with the Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX, USA. He has published over 360 journal articles and refereed conference papers. His research interests include cross-layer nanometer IC design for manufacturability, reliability, security, machine learning, and hardware acceleration, design/CAD for analog/mixed signal designs, and emerging technologies.

Prof. Pan has received numerous awards for his research contributions, including the SRC Technical Excellence Award, 18 Best Paper Awards, the NSF CAREER Award, and the IBM Faculty Award four times. He has served in many ACM/IEEE editorial boards and conference committees, including various leadership roles. He is a fellow of SPIE.

Michaela Brunner received the bachelor's and master's degrees in electrical and computer engineering from the Technical University of Munich, Munich, Germany, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree with the Chair of Security in Information Technology.

Her primary research interests are in the area of hardware security, sequential netlist reverse engineering, including gate-level netlist analysis, as well as netlist obfuscation, netlist encryption, and locking methods.

Ms. Brunner received the VDE Award 2018 from the Organization of German Electrical Engineers in South Bavaria for Her Master Thesis.

Georg Sigl received the Ph.D. degree in electrical engineering from the Technical University Munich, Munich, Germany, in 1992.

He held several positions in research and development with Siemens, Munich, and Infineon, Neubiberg, Germany. From 2000 to 2010, he was responsible for the development of new secure microcontroller platforms in the Chip Card and Security Division. Under his responsibility, two award winning platforms have been designed. In June 2010, he founded a New Chair with the Technical University of Munich for Security in Electrical Engineering and Information Technology. He is driving embedded security research as the Director of the Fraunhofer Research Institute for Applied and Integrated Security AISEC Munich.

Ulf Schlichtmann (Senior Member, IEEE) received the Dipl.-Ing. and Dr.-Ing. degrees in electrical engineering and information technology from the Technical University of Munich (TUM), Munich, Germany, in 1990 and 1995, respectively.

He is a Professor and the Head of the Chair of Electronic Design Automation with TUM. He joined TUM in 2003, following 10 years in industry. His current research interests include computer-aided design of electronic circuits and systems, with an emphasis on designing reliable and robust systems. He also addresses emerging technologies, such as lab-on-chip and photonics.