# Timing Resilience for Efficient and Secure Circuits

Grace Li Zhang[1], Michaela Brunner[2], Bing Li[1], Georg Sigl[2], Ulf Schlichtmann[1]
[1]Chair of Electronic Design Automation, Technical University of Munich (TUM), Germany
[2]Chair of Security in Information Technology, Technical University of Munich (TUM), Germany
Email: {grace-li.zhang, michaela.brunner, b.li, sigl, ulf.schlichtmann}@tum.de

**Abstract— In this paper, we will cover several techniques that can enhance the resilience of timing of digital circuits. Using post-silicon tuning components, the clock arrival times at flip-flops can be modified after manufacturing to balance delays between flip-flops. The actual delay properties of flip-flops will be examined to exploit the natural flexibility of such components. Wave-pipelining paths spanning several flip-flop stages can be integrated into a synchronous design to improve the circuit performance and to reduce area. In addition, with this technique, it cannot be taken for granted anymore that all the combinational paths in a circuit work with respect to one clock period. Therefore, a netlist alone does not represent all the design information. This feature enables the potential to embed wave-pipelining paths into a circuit to increase the complexity of reverse engineering. In order to replicate a design, attackers therefore have to identify the locations of the wave-pipelining paths, in addition to the netlist extracted from reverse engineering. Therefore, the security of the circuit against counterfeiting can be improved.**

## I. Introduction

Timing, one of the core performance metrics in digital circuits, indicates how fast a circuit can process data. Timing performance is evaluated by the maximum clock frequency which a circuit can operate with. This performance was improved significantly owing to the advancement of manufacturing technology and design methodology. But this improvement cannot be maintained any more and the timing performance has stagnated in recent years.

With manufacturing technology reaching a nanoscale level, the size of transistors becomes smaller and smaller. On the one hand, this shrinking size brings smaller propagation delays for combinational gates, and thus smaller clock periods for digital circuits. On the other hand, it results in undesirable side-effects. For example, the increasing manufacturing variations cause variations of physical parameters, e.g., gate length. Such variations in turn cause electrical parameters, e.g., $V_{th}$, to differ from their nominal specifications. Consequently, combinational gates exhibit different delays in different chips after manufacturing. Recent work shows that even FinFETs exhibit large process variations [1], [2].

To take the impact of process variations into account, worst-case timing analysis has been deployed in the IC (integrated circuit) industry in the past several decades. In this method, each process parameter is set to the value in the worst condition independently without considering their correlations. With this setting, the worst-case timing performance can be evaluated. However, the timing performance evaluated with this method is extremely pessimistic, leading to an overdesign that wastes design effort. In recent years, various methods have been explored to deal with process variations, e.g., analyzing timing under process variations [3]–[16].

Another challenge, circuit aging, degrades device characteristics under stress over time [17]–[19], and thus timing performance of circuits is lowered correspondingly. To analyze aging effects, timing model and algorithms on gate level [20]–[23] have been introduced in recent years.

To overcome the challenges described above, we have to reexamine the concepts in the traditional timing paradigm. For example, signals propagating inside a combinational block terminate at the flip-flops and do not propagate further until the next clock edge arrives. Consequently, timing of a digital circuit can be defined with respect to one clock period and isolated within individual flip-flop stages. Such a strict timing definition reduces the design effort significantly. However, it affects timing performance negatively due to the barriers of flip-flops. It is desirable to develop a new timing concept to improve the timing performance of circuits.

In this paper, several techniques that can enhance the resilience of timing of digital circuits are summarized. A clock tuning technique with tunable buffers to balance delays between flip-flop stages by adjusting the clock arrival times at flip-flops after manufacturing is investigated in Section II. The timing characteristics of flip-flops can also be exploited to alleviate the impact of process variations, as described in Section III. Thereafter, wave-pipelining is integrated into circuits to improve the timing performance beyond the limit in the traditional timing paradigm in Section IV. This concept can also be used to enhance the security of digital circuits, as described in Section V.

## II. Clock Tuning after Manufacturing

To counter process variations, clock tuning with tunable buffers can be used to modify the timing properties of flip-flops for each manufactured chip individually [8], [10]. We use Fig. 1 to explain the concept of this method. In this figure, combinational paths, represented by inverters, connect flip-flops. The delays of the corresponding combinational paths are shown next to the inverters. Because of process variations, path delays are uncertain during the design phase and can be treated as statistical variables. After manufacturing, they are fixed in each chip, so that clock skew tuning can be applied to counter process variations.

If all tunable buffers in Fig. 1 would have zero delays, the minimum clock period the circuit can achieve is 8 units. It can be reduced from 8 to 5.5 units if the tunable buffers are used to adjust the clock arrival times at flip-flops. For instance, the buffer inserted at F2 is configured with a negative delay of -2.5 units, which shifts the clock edge to arrive 2.5 units earlier



978-1-7281-4123-7/20/$31.00 

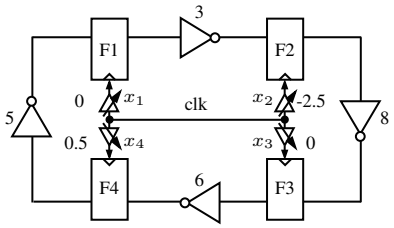


Figure 1: Reduction of the minimum clock period with clock tuning buffers.

at F2. This shift ensures that the path between F2 and F3 still has 8 time units to finish logic computations with a clock period of 5.5 units. Although this shift reduces the time for signal propagation along the path between F1 and F2, there are still no timing violations at F2. Since a reference clock signal is used to define buffer delays, negative delays can be achieved by reducing the length from the original clock path to flip-flops.

Timing constraints for a circuit with tunable buffers are illustrated in Fig. 2, where tunable buffers are attached to flip-flops $i$ and $j$. The active clock edge is assumed to appear at reference time 0. Because of the tunable buffers, the arrival times of the active clock edges at the two flip-flops become $x_i$ and $x_j$, respectively. To avoid timing violations, the following constraints must be satisfied

$$x_i + \overline{d}_{ij} \leq x_j + T - s_j \tag{1}$$

$$x_i + \underline{d}_{ij} \geq x_j + h_j \tag{2}$$

where $x_i$ and $x_j$ are the delays of the tunable buffers, $\overline{d}_{ij}$ and $\underline{d}_{ij}$ are the largest and the smallest delays of the paths between the flip-flops, $T$ is the specified clock period with which the circuit operates, and $s_j$ and $h_j$ are the setup and hold times of the flip-flop $j$.

Since tunable buffers incur area overhead, tunable buffers can only be configured to a limited delay range. For a buffer $i$, the delay range is written as follows

$$r_i \leq x_i \leq r_i + \tau_i \tag{3}$$

where $r_i$ and $\tau_i$ are constants. They can be determined with methods such as [24]. Due to the implementation of buffers, $x_i$ may only take discrete values.

To apply post-silicon clock tuning, two challenges have to be overcome. First, during the design phase, those locations of tunable buffers that can improve yield as much as possible should be determined. Second, after manufacturing, delay test is required to identify combinational paths which do not meet timing. The cost of this delay test should be minimized.

The goal of buffer insertion during the design phase is to find the locations of buffers that are effective in improving yield. However, the relation between the yield and the buffer locations can not be established directly, since the path delays in (1)–(2) are statistical during the design phase. In this scenario, the insertion problem becomes a statistical optimization problem. To reduce the complexity, a certain number of representative samples are used to emulate manufactured chips, for which statistical delays become fixed. Consequently, the statistical optimization problem is transformed into a

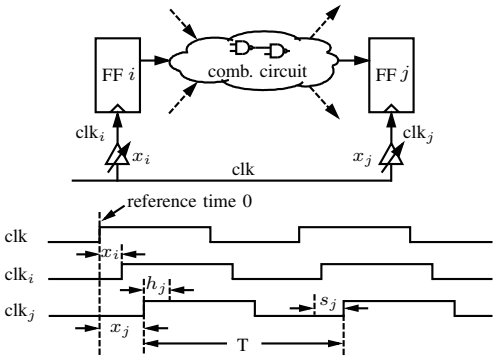


Figure 2: Timing with tunable buffers.

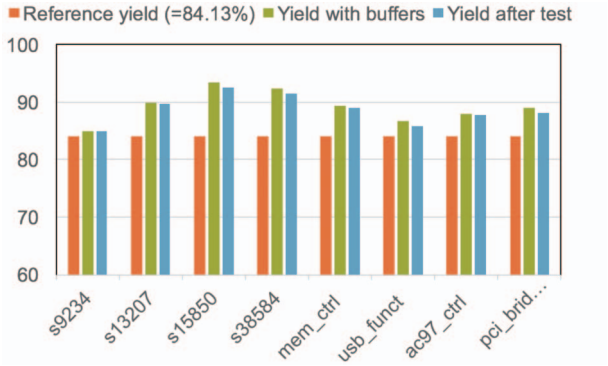


Figure 3: Results of yield improvement.

deterministic optimization problem. With sufficient samples, the yield of the circuits can be defined with respect to the buffer locations. With this relation, the locations of buffers that are important to yield improvement can be determined by maximizing yield. Details can be found in [25].

After manufacturing, the delays of combinational paths are fixed. If a chip has timing failures, tunable buffers can be adjusted to rescue this chip. The configuration values of such buffers in failed chips are determined by solving a problem formulation with the constraints (1)–(3). To find a viable solution, the path delays in the manufactured chips have to be evaluated. However, this task is very challenging. On the one hand, relatively accurate delays are desirable, so that buffers can be configured properly to improve yield. On the other hand, the cost incurred by this delay test must remain low to avoid that the yield improvement with post-silicon tuning becomes meaningless because of the high test cost.

To reduce the test costs, two techniques are applied. First, the correlation of path delays can be used to reduce the number of tested paths. Only the delays of a certain number of representative paths need to be tested. These results can be used together with the path correlations to estimate the remaining path delays. Second, existing tunable buffers can be used to align path delays, so that the delay information of multiple paths can be obtained simultaneously in one test iteration. Details can found in [26].

In the experiments, the number of buffers is bounded to be smaller than 1% of the number of flip-flops. The reference yield is 84.13% when the specified clock period is the sum of the mean value and the standard deviation of the statistical clock period. Fig. 3 demonstrates the yield improvement with respect to the reference yield [25], [26]. The green bars show

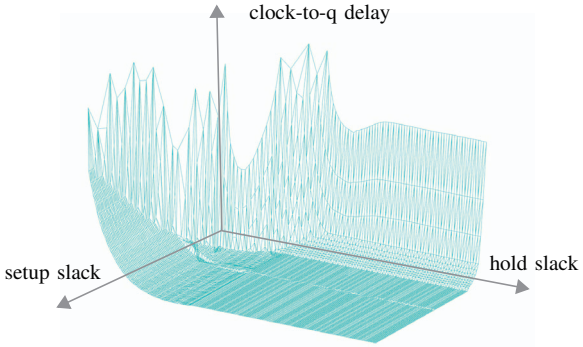


Figure 4: The relation between the flip-flop delay, setup and hold slacks.

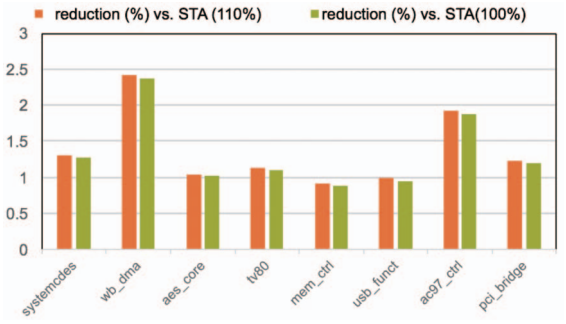


Figure 5: Clock period reduction of the piecewise model compared with STA.

that the yield is improved with post-silicon tuning when delays are assumed to be tested accurately. The blue bars show that the yield improvement is slightly degraded after delay test with the proposed method.

## III. Setup/Hold Time Interdependency

Facing challenges from process variations, we have to reexamine the traditional definitions of timing concepts. For example, the clock-to-q delay of a flip-flop, abbreviated as the flip-flop delay as follows, and both setup time and hold time are assumed to be constant values in static timing analysis (STA). In addition, if the setup and hold time constraints are satisfied, flip-flops are considered to function correctly. In reality, flip-flops may still latch data correctly when the setup and hold constraints are violated in some degree, though the delays of flip-flops may increase [27]. Consequently, setup and hold time violations do not necessarily lead to the malfunction of the flip-flop. If these flexible characteristics can be used, delay differences between flip-flop stages can be reduced. Consequently, the effects of process variations can be mitigated without incurring area overhead.

To exploit the flexibility of flip-flops, we define *setup slack* to be the time gap between the change of a signal arriving at a flip-flop and the active clock edge. Similarly, we define *hold slack* to be the time gap between the active clock edge and the change of a signal. The interdependency of the flip-flop delay and the two slacks is illustrated in Fig. 4. In this figure, it is clear that for large setup and hold slacks the delay surface is flat. When these slacks become small, the flip-flop delay increases, and eventually reaches points where the flip-flop becomes metastable and might not latch data correctly. The simplification in STA does not exploit the region where setup and hold slacks are smaller than the values defined in the cell library, leading to an underestimation of timing performance potentially.

To exploit the flexible timing characteristics of a flip-flop, we have to overcome two challenges. First, a relatively accurate model of the three-dimensional delay surface in Fig. 4 should be determined. Second, a timing optimization algorithm is required to evaluate the real maximum clock frequency of a circuit with the accurate delay model. Previous studies [27]–[30] either did not solve the problem considering the three-dimensional delay surface, or cannot provide a high-quality solution.

To model the three-dimensional surface, we partition it into small polygons, e.g., triangles and rectangles. After the delay surface is approximated with these polygons, the real maximum clock frequency of a circuit is calculated with a piecewise ILP model. Details can be found in [31].

To demonstrate the resulting timing performance improvements, the clock periods which are calculated with the piecewise model were compared with the results from traditional STA. The comparisons are shown in Fig. 5 [31]. The reduced clock periods only result from the accurate consideration of the timing parameter relation in the piecewise linear model.

## IV. Virtual Synchronization with Delay Elements

In the traditional timing paradigm, signal propagations are synchronized with flip-flops and cannot travel through them except at an active clock edge. However, flip-flops have inherent delays and require setup times, so that they can only slow down signal propagations, but never speed them up. Consequently, if flip-flops along critical paths are removed, signal propagations along such paths are accelerated. In addition, delay imbalances between flip-flop stages are exploited automatically.

Fig. 6 illustrates a scenario where a flip-flop is removed to improve timing performance. The smallest possible clock period of a circuit is defined by the largest delay between two flip-flops. In Fig. 6(a), the largest delay is equal to 21 units, considering a flip-flop delay of 3 units and a setup and hold time of both 1 unit. To reduce the smallest clock period, combinational gates with smaller delays can be chosen from the library. The cost of such gate sizing is an increase in area. The resulting circuit is illustrated in Fig. 6(b), where the largest delay is now equal to 16 units. In addition, the flip-flop F3 can be moved leftwards to improve timing performance further, as shown in Fig. 6(c). The largest delay after retiming is now equal to 11 units, which cannot be reduced anymore in the traditional timing paradigm.

To further improve timing performance, flip-flops can be removed from the circuits. Fig. 6(d) illustrates an example to explain this concept. In this figure, the flip-flop F6 is removed from the circuit in Fig. 6(c). As a result, two data waves propagate along the path between F2 and F4 and the path between F2 and F3 simultaneously. To guarantee that the functionality does not change, the arrival time of signals from F2 at F3 and F4 should be greater than one clock period

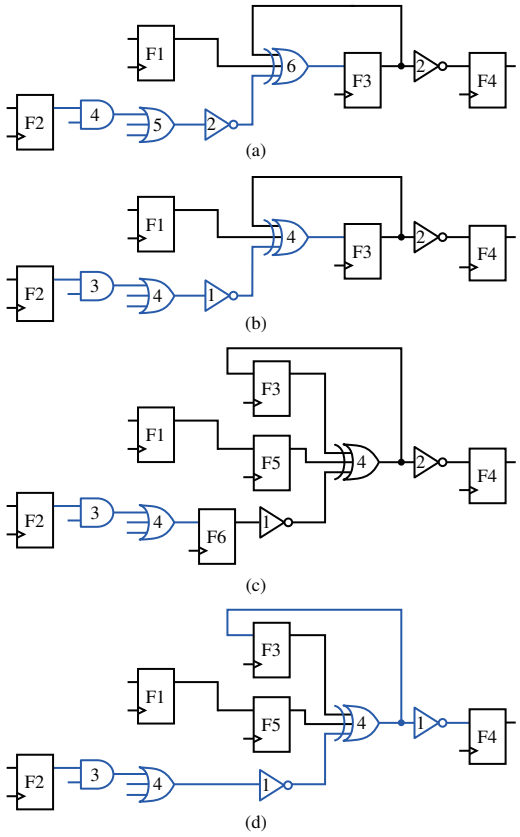


Figure 6: The concept of virtual synchronization. The gate delays are indicated on the gates. (a) The original circuit. (b) The circuit with gate sizing. (c) The circuit with retiming and gate sizing. (d) The circuit with removing F6 and gate sizing.

and less than two clock periods. The largest path delay of the circuit in Fig. 6(d) can be reduced to 8.5 units, half of the largest delay between F2 and F4. This resulting, smallest clock period is significantly lower than the limit in the traditional timing paradigm.

To apply such wave-pipelining to reduce the clock period, we first remove all sequential components such as flip-flops from the circuit. As a result, signals along critical paths will be accelerated. However, this can lead to incorrect arrival times of signals along fast paths, for example an earlier arrival compared with the one in the original circuit. In addition, there is a loss of synchronization for signals along combinational loops due to iterative travels within the loops. Therefore, if wave-pipelining is applied, the challenge is to slow down signals propagating along fast paths and loops. For this purpose, different circuit components are used as delay elements [32]. They are combinational delay elements (e.g. buffers or a chain of inverters), and sequential delay elements: latches and flip-flops. Each of them has its individual delay characteristic.

To improve timing performance with virtual synchronization, the number of delay elements inserted into a circuit to slow down fast signals and loops should be as small as possible. The insertion of delay elements first finds the locations where sequential delay elements are necessary by emulating their delaying effects. Afterwards, these locations are refined by incorporating the inherent delays of these

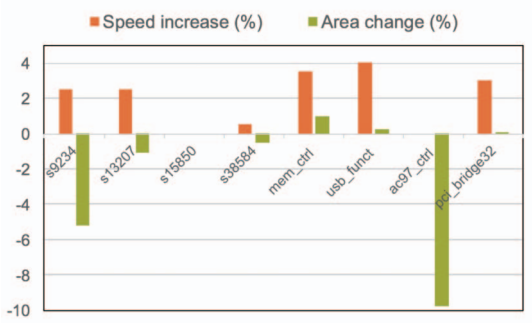


Figure 7: Speed increase and area results of VirtualSync compared to ideally balanced design.

sequential delay elements. Details of this approach, called VirtualSync, can be found in [32].

The comparisons of speed increase and area change with ideally balanced design combining gate sizing and retiming are shown in Fig. 7 [32]. As this figure shows, the timing performance in most circuits exceeds the limit achievable in the traditional timing paradigm. In addition, the area can be reduced because of the removal of flip-flops in most cases.

## V. Timing Camouflage for Netlist Security

Due to globalization, the supply chain of ICs becomes distributed, making them vulnerable to counterfeiting, product piracy and various attacks, like for example the insertion of hardware Trojans. One serious counterfeiting threat is the chip-level reverse engineering of gate-level netlists by attackers which can then for example be used to produce illegal chips. The chip-level reverse engineering of an authentic chip includes among other steps the depackaging, the extraction of the different layers and their images, as well as the recognition of combinational and sequential gates and their connections [33]. With EDA toolchains, attackers can process recognized netlists to replicate chips illegally. To enhance netlist security, different mechanisms were introduced to prevent reverse engineering or the usage of the extracted netlists. Those countermeasures include for example locking methods, e.g., [34], [35], camouflaging, e.g., [36], [37] or split manufacturing, e.g., [38]. Locking methods provide a corrupted output or functionality if a wrong secret key is applied. Camouflaging hides the actual functionality of the extracted gates and split manufacturing separates the front-end-of-line and back-end-of-line manufacturing processes. More recent countermeasure techniques also incorporate timing information. This is either done as corruption outcome, like a decreased circuit performance for a wrong applied key [39], or as locking strategy aid or replacement [37], [40], [41]. For example in [40], tunable delay buffers are added and controlled by secret key bits, in [41], finite state machine transitions are based on applied circuit frequencies, or in [37], cells with basically the same layout geometry but different delays hide the actual circuit function. Next to reverse engineering based attacks, also side-channel, fault and probing attacks [42] can reveal circuit specific information. Possible countermeasures are for example introduced in [13], [43].

The reverse engineering flow described above to duplicate chips with the aid of state of the art EDA toolchains is only

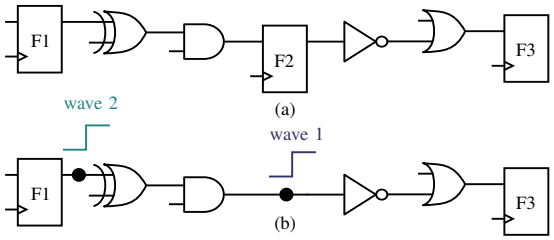


Figure 8: Concept of Timing Camouflage. (a) Single-period clocking; (b) Wave-pipelining between F1 and F3.

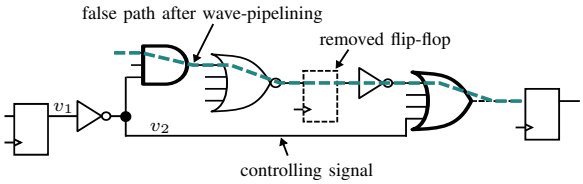


Figure 9: The false path with wave-pipelining is formed with two true single-period paths.

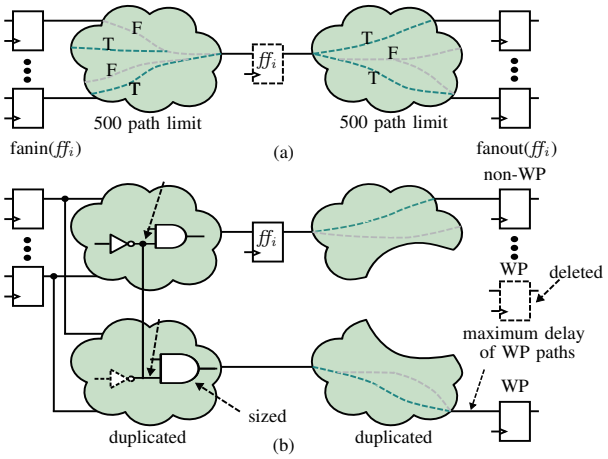


Figure 10: Wave-pipelining for netlist security. (a) Removal of the flip-flop $ff_i$. "T" and "F" represent true and false paths. (b) Replicated logic block and gate sizing.

effective when all combinational paths work with respect to one clock period. Fig. 8(a) illustrates an example of such a single-period clocked circuit. In this figure, a partial sequential circuit is shown which consists of three flip-flops and four combinational gates. In the traditional digital paradigm, timing is only verified between pairs of flip-flops. Consequently, the unsecured netlist in Fig. 8(a) is sufficient for attackers to reproduce the original design, since they only have to identify the combinational gates, sequential components and how they connect with each other to recover the original netlist.

To enhance netlist security, we invalidate the assumption that a netlist is sufficient to reproduce the original design by incorporating wave-pipelining paths into the circuit. For instance, the circuit in Fig. 8(b) [44] can be constructed by removing F2 in Fig. 8(a). After the removal of the flip-flop, the combinational path between F1 and F3 has 2 data waves propagating along it simultaneously. If the first data wave is not flushed away by the second data wave before it is latched by F3, the functionality of the circuit is still guaranteed.

When attackers face the circuit incorporated with wave-pipelining in Fig. 8(b), they have to recognize the number of data waves along combinational paths in addition to gate types and connections. If they wrongly assume the existence of only one data wave and therefore handle the netlist with the standard EDA flow, F3 latches data one clock period earlier than the original circuit. Consequently, the recovered circuit will not work correctly due to the loss of synchronization. In order to detect the existence of wave-pipelining paths, attackers have to invest additional effort to obtain timing information of combinational paths. One possibility would be to deploy testing technique to determine path delays of authentic chips bought from the market [45]–[47]. To prevent attackers from successfully applying such testing technique, wave-pipelining false paths are introduced, because they are unsensitizable by testing under the single-period clocking. Fig. 9 illustrates an example of a false path with wave-pipelining. After the flip-flop in the middle is removed, a path with wave-pipelining is formed. When this path is considered to work with single-period clocking, a signal change at the start of this wave-pipelining path is blocked before it arrives a flip-flop. If the value of signal $v_2$ is '0', signal propagations are blocked at the first AND gate. If the value is '1', signal propagations are blocked at the last OR gate. Consequently, the delay of this wave-pipelining path cannot be tested with one clock period, making it a false path, and therefore robust against traditional testing. In addition, the number of real false paths in digital circuits is very large, about 75% of the number of combinational paths [48]. Consequently, it is very difficult for attackers to distinguish false paths with wave-pipelining from real false paths working within one clock period.

When applying the wave-pipelining technique to secure a circuit, the original functionality of a circuit should be maintained. To achieve this goal, the following timing constraints for the wave-pipelining paths should be satisfied:

1) Their delays have to be greater than one clock period to avoid the early latching of signals.
2) Their delays have to be less than two clock periods.

To achieve the construction of wave-pipelining paths in a circuit shown in Fig. 10(a), the flip-flop in the middle can be removed. Unfortunately, the removal of the flip-flop turns all paths connected with it into wave-pipelining paths. Since there are a lot of short paths leftwards and rightwards, if they are connected directly, a lot of short paths with small delays are generated. The small delays of these paths might violate the wave-pipelining constraints. To overcome the challenges described above, we replicate the combinational logic gates as well as the flip-flop at the end of the wave-pipelining paths in the original circuit, as illustrated in Fig. 10(b). To reduce the resource usage incurred by the replication, we only replicate the combinational logic along the wave-pipelining paths on the right side of $ff_i$. Afterwards, the flip-flop at the end of the wave-pipelining paths in the original circuit is deleted and the combinational gates that have no connection with any flip-flops are removed in the original circuit, as shown in Fig. 10(b). To guarantee the correct functionality, on the left side of $ff_i$, the whole combinational logic is replicated first. Afterwards, we try to reduce the resource usage by using the original logic gates. For instance, the inverter in the replicated circuit can be removed by connecting one of the input pins of

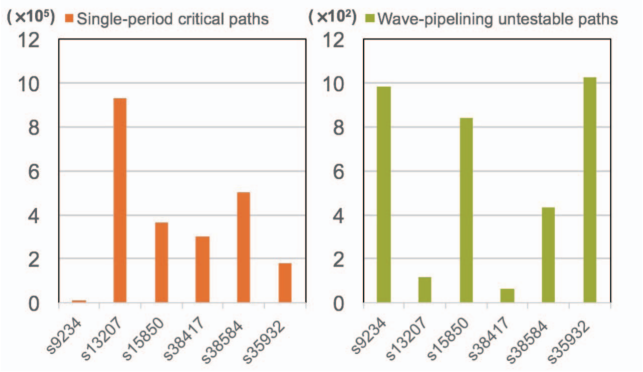


Figure 11: Results of constructing wave-pipelining paths.

the AND gate in the replicated circuit with the corresponding pin in the original circuit. In addition, we also size gates and insert buffers to extend the delays of wave-pipelining paths to ensure that wave-pipelining constraints can be satisfied. Details can be found in [44].

Fig. 11 demonstrates the results of wave-pipelining construction [44]. The left figure demonstrates the total number of suspicious single-period true paths that force attackers to perform testing. The right figure shows the total number of false paths with wave-pipelining that are untestable. To replicate chips, attackers need to recognize the constructed false paths with wave-pipelining from the original false paths working within one clock period, whose number is about 75% of the number of combinational paths in the original circuit [48]. Therefore, an attack on this camouflage technique is still challenging.

## VI. Conclusion

In this paper, we present several methods to enhance the resilience of timing of digital circuits. Post-silicon clock tuning deploys tunable buffers to adjust the clock skews to flip-flops individually for each chip after manufacturing. The natural timing flexibility of flip-flops is exploited to mitigate the effects of process variations. Since flip-flops are barriers of timing performance, they are removed to incorporate wave-pipelining into circuits. The timing performance of the resulting circuit can break through the limit of the traditional timing paradigm. In addition, this concept can also be used to enhance the security of digital circuits. This technique, called timing camouflage, significantly improves the resilience of circuits against counterfeiting.

## References


[1] V. B. Kleeberger, H. Graeb, and U. Schlichtmann, "Predicting future product performance: modeling and evaluation of standard cells in FinFET technologies," in *Proc. Design Autom. Conf.*, 2013, pp. 33:1–33:6.

[2] S. Karapetyan, V. B. Kleeberger, and U. Schlichtmann, "FinFET-based product performance: Modeling and evaluation of standard cells in FinFET technologies," *Microelectronics Reliability*, vol. 61, pp. 30–34, 2016.

[3] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer, "Statistical timing analysis: From basic principles to state of the art," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 4, pp. 589–607, 2008.

[4] C. Visweswariah, K. Ravindran, K. Kalafala, S. Walker, and S. Narayan, "First-order incremental block-based statistical timing analysis," in *Proc. Design Autom. Conf.*, 2004, pp. 331–336.

[5] B. Li, N. Chen, M. Schmidt, W. Schneider, and U. Schlichtmann, "On hierarchical statistical static timing analysis," in *Proc. Design, Autom., and Test Europe Conf.*, 2009, pp. 1320–1325.

[6] B. Li, N. Chen, Y. Xu, and U. Schlichtmann, "On timing model extraction and hierarchical statistical timing analysis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 3, pp. 367–380, 2013.

[7] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge, "Razor: A low-power pipeline based on circuit-level timing speculation," in *Proc. Int. Symp. Microarch.*, 2003, pp. 7–18.

[8] D. Tadesse, J. Grodstein, and R. I. Bahar, "AutoRex: An automated post-silicon clock tuning tool," in *Proc. Int. Test Conf.*, 2009, pp. 1–10.

[9] B. Li, N. Chen, and U. Schlichtmann, "Fast statistical timing analysis of latch-controlled circuits for arbitrary clock periods." in *Proc. Int. Conf. Comput.-Aided Des.*, 2010, pp. 524–531.

[10] 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, 2015.

[11] R. Kumar, B. Li, Y. Shen, U. Schlichtmann, and J. Hu, "Timing verification for adaptive integrated circuits," in *Proc. Design, Autom., and Test Europe Conf.*, 2015, pp. 1587–1590.

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

[13] A. Herrmann, M. Weiner, M. Pehl, and H. Gräb, "Bringing analog design tools to security: Modeling and optimization of a low area probing detector," in *Int. Conf. on Syn., Mod., Ana. and Simu. Methods and Appl. to Circuit Des.*, 2018.

[14] G. L. Zhang, B. Li, and U. Schlichtmann, "EffiTest: Efficient delay test and statistical prediction for configuring post-silicon tunable buffers," in *Proc. Design Autom. Conf.*, 2016, pp. 60:1–60:6.

[15] B. Yigit, G. L. Zhang, B. Li, and U. Schlichtmann, "Application of machine learning methods in post-silicon yield improvement," in *Proc. Int. System-on-Chip Conf. (SOCC)*, 2017, pp. 243–248.

[16] R. Kumar, B. Li, Y. Shen, U. Schlichtmann, and J. Hu, "Timing verification for adaptive integrated circuits," in *Proc. Design, Autom., and Test Europe Conf.*, 2015, pp. 1587–1590.

[17] A. H. Baba and S. Mitra, "Testing for transistor aging," in *Proc. VLSI Test Symp.*, 2009, pp. 215–220.

[18] V. B. Kleeberger, M. Barke, C. Werner, D. Schmitt-Landsiedel, and U. Schlichtmann, "A compact model for NBTI degradation and recovery under use-profile variations and its application to aging analysis of digital integrated circuits," *Microelectronics Reliability*, vol. 54, no. 6–7, pp. 1083–1089, 2014.

[19] H. Amrouch, B. Khaleghi, A. Gerstlauer, and J. Henkel, "Reliability-aware design to suppress aging," in *Proc. Design Autom. Conf.*, 2016, pp. 12:1–12:6.

[20] D. Lorenz, G. Georgakos, and U. Schlichtmann, "Aging analysis of circuit timing considering NBTI and HCI," in *Int. On-Line Testing Symp. (IOLTS)*, 2009, pp. 3–8.

[21] D. Lorenz, M. Barke, and U. Schlichtmann, "Efficiently analyzing the impact of aging effects on large integrated circuits," *Microelectronics Reliability*, vol. 52, no. 8, pp. 1546–1552, 2012.

[22] D. Lorenz, M. Barke, and Schlichtmann, "Monitoring of aging in integrated circuits by identifying possible critical paths," *Microelectronics Reliability*, vol. 54, no. 6-7, pp. 1075–1082, 2014.

[23] D. Lorenz, M. Barke, and U. Schlichtmann, "Aging analysis at gate and macro cell level," in *Proc. Int. Conf. Comput.-Aided Des.*, 2010, pp. 77–84.

[24] J. Tsai, L. Zhang, and C. C.-P. Chen, "Statistical timing analysis driven post-silicon-tunable clock-tree synthesis," in *Proc. Int. Conf. Comput.-Aided Des.*, 2005, pp. 575–581.

[25] G. L. Zhang, B. Li, J. Liu, Y. Shi, and U. Schlichtmann, "Design-phase buffer allocation for post-silicon clock binning by iterative learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 2, pp. 392–405, 2018.

[26] 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, 2019.

[27] Y.-M. Yang, K. H. Tam, and I. H.-R. Jiang, "Criticality-dependency-aware timing characterization and analysis," in *Proc. Design Autom. Conf.*, 2015, pp. 167:1–167:6.

[28] A. Jain and D. Blaauw, "Slack borrowing in flip-flop based sequential circuits," 2005, pp. 96–101.

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

[30] A. B. Kahng and H. Lee, "Timing margin recovery with flexible flip-flop timing model," in *Proc. Int. Symp. Quality Electron. Des.*, 2014, pp. 496–503.

[31] 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. 100:1–100:8.

[32] 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.

[33] M. Fyrbiak, S. Strauß, C. Kison, S. Wallat, M. Elson, N. Rummel, and C. Paar, "Hardware reverse engineering: Overview and open challenges," in *2017 IEEE 2nd International Verification and Security Workshop (IVSW)*, 2017, pp. 88–94.

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

[35] R. S. Chakraborty and S. Bhunia, "Harpoon: an obfuscation-based soc design methodology for hardware protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1493–1502, 2009.

[36] B. Shakya, H. Shen, M. Tehranipoor, and D. Forte, "Covert gates: Protecting integrated circuits with undetectable camouflaging," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 86–118, 2019.

[37] D. Darmon, A. Klein, Y. Salmon, A. Grabovsky, and R. Attia, "Timing based camouflage circuit," Apr. 16 2019, US Patent 10,262,956.

[38] R. Jarvis and M. Mcintyre, "Split manufacturing method for advanced semiconductor circuits," Mar. 27 2007, US Patent 7,195,931.

[39] L. Li and H. Zhou, "Structural transformation for best-possible obfuscation of sequential circuits," in *IEEE Proc. Int. Symp. Hardware Ori. Security & Trust*, 2013, pp. 55–60.

[40] Y. Xie and A. Srivastava, "Delay locking: Security enhancement of logic locking against ic counterfeiting and overproduction," in *Proc. Design Autom. Conf.*, 2017, p. 9.

[41] K. Juretus and I. Savidis, "Time domain sequential locking for increased security," in *Proc. Int. Symp. Circuits and Syst.*, 2018, pp. 1–5.

[42] B. Selmke, K. Zinnecker, P. Koppermann, K. Miller, J. Heyszl, and G. Sigl, "Locked out by latch-up? An empirical study on laser fault injection into Arm Cortex-M processors," in *Workshop on Fault Diagno. and Toler. in Cryp.*, 2018.

[43] M. Weiner, S. Manich, R. Rodríguez-Montañés, and G. Sigl, "The low area probing detector as a countermeasure against invasive attacks," *IEEE Trans. VLSI Syst.*, vol. 26, no. 2, pp. 392–403, Feb 2018.

[44] 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., and Test Europe Conf.*, 2018.

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

[46] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, "SARLock: SAT attack resistant logic locking," in *IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2016, pp. 236–241.

[47] 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.

[48] 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.