Timing with Virtual Signal Synchronization for Circuit Performance and Netlist Security

Grace Li Zhang, Bing Li, Ulf Schlichtmann
Chair of Electronic Design Automation, Technical University of Munich (TUM), Munich, Germany
Email: {grace-li.zhang, b.li, ulf.schlichtmann}@tum.de

***Abstract*—In digital circuit design, sequential components, e.g., flip-flops, are used to synchronize signal propagations using a global clock signal. With this design style, logic blocks are isolated by flip-flop stages, so that timing constraints can be addressed between pairs of flip-flops. Accordingly, the minimum clock period is determined by the maximum combinational delay between flip-flops. This design style reduces design efforts significantly. However, the timing performance of digital circuits might be affected negatively because sequential components can only delay signal propagations instead of accelerating them. In addition, the assumption that all combinational paths work within a single clock period indicates that the netlist carries all the design information, which makes ICs vulnerable to counterfeiting. In this paper, we demonstrate two techniques to break the confines of the traditional timing paradigm: VirtualSync and TimingCamouflage. With these techniques, circuit performance can be pushed even beyond the limit of the traditional timing paradigm and the netlist security can be enhanced.**

I. Introduction

In traditional digital circuits, logic design and timing verification are separated. This design style is made possible by using sequential components, e.g., flip-flops, to block signal propagations along logic gates that actually process data. Consequently, logic blocks are isolated by flip-flops, so that timing verification is only needed between all pairs of flip-flops. This fully synchronous design style reduces design efforts significantly. However, it potentially leads to timing performance degradation. On one hand, sequential components such as flip-flops have clock-to-q delays, which become a part of the delays of combinational paths driven by the corresponding flip-flops. They also force signals to arrive setup time before the active clock edge, which consumes a further part of the timing budget from critical paths. On the other hand, delay compensation between consecutive flip-flop stages cannot be achieved because flip-flops block signal propagations instead of allowing them to pass through.

Figure 1 illustrates the scenarios where the timing performance in the traditional paradigm is restricted by the barriers of flip-flops. In Fig. 1(a), the circuit achieves a minimum clock period of 21 units. To reduce the clock period, logic gates can be sized to achieve smaller delays, so that signal propagations on the critical paths of the circuits can be accelerated. Fig. 1(b) shows the circuit after gate sizing with a smaller minimum clock period of 16 units. Retiming can be deployed to reduce the clock period further by moving F3 in Fig. 1(b) leftwards. Fig. 1(c) shows the circuit after retiming, whose functionality is the same as that in Fig. 1(b). The minimum clock period in Fig. 1(c) is 11 units, already the limit of timing performance in the traditional timing paradigm.

If the confines of the traditional timing paradigm are broken by removing the barriers of flip-flops, the timing performance can be pushed further. Fig. 1(d) illustrates this concept, where F6 is removed from Fig. 1(c). If correct data is latched by F3 and F4 by guaranteeing that the signal from F2 reaches F3 and F4 after T and before $2T$, the functionality of the circuit is still maintained. The largest path delay is 16. Together with the setup time of 1 unit, the minimum clock period can be reduced further to (16+1)/2=8.5, 22.7% lower than the optimum performance in the traditional single-period clock paradigm.

The technique of allowing signals to span across flip-flop stages without a flip-flop blocking them can automatically balance timing slacks between successive flip-flop stages. Therefore, it alleviates the effect of process variations and aging, which are two factors affecting the IC design flow significantly. To counter process variations, previous methods such as post-silicon tuning [1], [2], [3], [4], [5], [6], [7] are deployed to tune circuits after manufacturing with respect to the effect of process variations. The technique to boost circuit performance until timing errors occur has been explored in the Razor method [8], [9], [10], [11]. In addition, the interdependency between setup and hold time has been considered to exploit the performance potential of flip-flops [12], [13], [14], [15]. To evaluate aging effects such as Negative Bias Temperature Instability (NBTI), Hot Carrier Injection (HCI), etc, timing models for aging analysis have been proposed [16], [17], [18], [19]. Among these methods, the AgeGate model [20], [21], [22] can be integrated into standard timing signoff flows directly [23]. The methods above dealing with process variations and aging incur additional overhead in circuit area, design and analysis complexity. This overhead can be reduced by removing the barriers of flip-flops.

The traditional timing paradigm also increases the security risk for digital circuits. In this paradigm, all combinational paths between pairs of flip-flops work within one clock period, indicating that the netlist carries all the design information. Therefore, attackers can reconstruct the original netlist to counterfeit chips with the recognized netlist from reverse engineering.

In this paper, we will discuss two techniques that break the confines of the traditional timing paradigm to improve the timing performance of digital circuits and to enhance circuit security against counterfeiting.

II. VirutalSync Timing Model

A. Sequential and Combinational Components as Delay Units

To achieve multiple waves on combinational paths, all sequential components, e.g., flip-flops, are firstly removed from the circuit under optimization. Consequently, signals

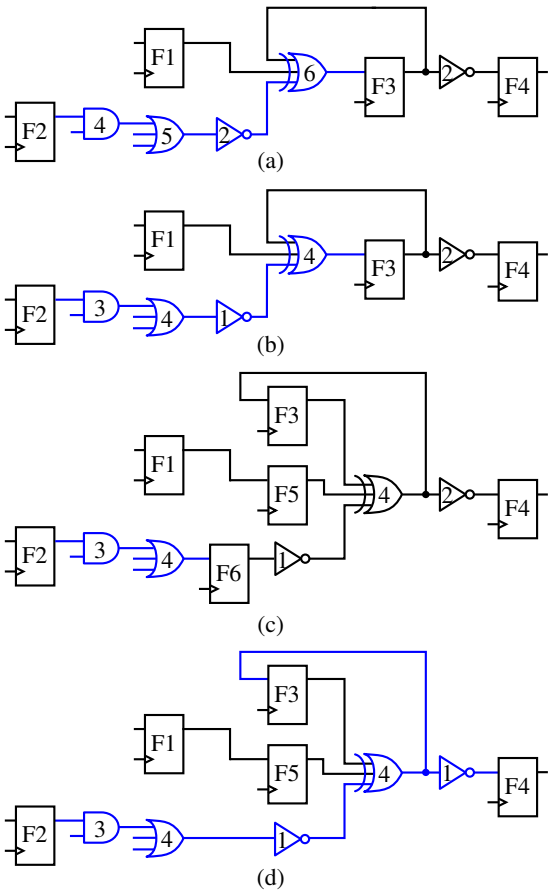

Fig. 1: Timing optimization methods. Delays of logic gates are shown on the gates. The clock-to-q delay (t_{cq}), setup time (t_{su}) and hold time (t_h) of a flip-flop are 3, 1 and 1, respectively. (a) Original circuit. (b) Sized circuit. (c) Circuit after retiming. (d) Circuit after optimization using VirtualSync.

across fast paths may arrive at flip-flops earlier than specified, leading to a loss of logic synchronization. Furthermore, a combinational loop must have a sequential component to guarantee logic synchronization after signals travel across it many times. Therefore, we need to delay the signal propagation on fast paths and combinational loops. To achieve this signal blocking, combinational gates such as buffers, flip-flops, and latches can be used as delay units. Fig. 2 illustrates the different delay characteristics of the three types of components, where ***input gap*** refers to the difference of arrival times of two signals at a delay unit, and ***output gap*** represents the difference between their arrival times after they pass through the unit.

In Fig. 2(a), a combinational delay unit such as a buffer has a linear delaying effect, indicating that there is no change in the absolute gap between arrival times of signals through short and long paths after passing through a buffer.

A flip-flop, as a sequential delay unit, has a different behavior, as shown in Fig. 2(b). No matter when two signals arrive at the input of a flip-flop, the departure time after passing through a flip-flop is the same. These characteristics are very useful to block signals on short paths when the delays of short and long paths are very different after all sequential components are removed.

Another sequential delay unit, a level-sensitive latch, has

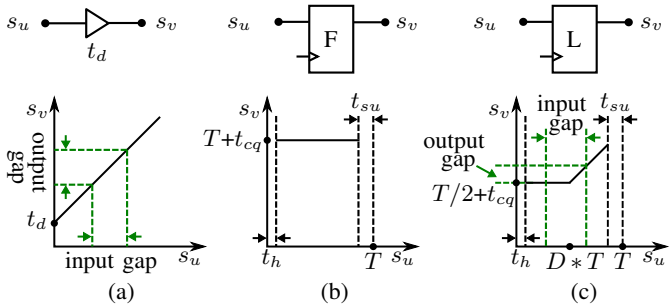

Fig. 2: Properties of delay units. (a) Linear delaying effect of a combinational delay unit. (b) Constant delaying effect of a flip-flop. (c) Piecewise delaying effect of a latch.

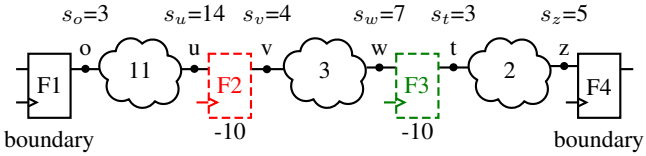

Fig. 3: Concept of relative timing references. Clock period T=10.

piecewise delaying effect, as shown in Fig. 2(c). The output gap of two input signals after passing through a non-transparent latch is reduced to zero. The output gap remains unchanged after a transparent latch is passed through. However, if the fast signals pass through a non-transparent latch and the slow signals pass through the latch when it is in transparency, the output gap takes a value between the two extreme values, as shown in Fig. 2(c). This property is very flexible to block signal propagations along critical paths where fast signals need to be delayed and slow signals should not be affected.

B. Relative Timing References

After removing sequential components from a circuit, signals may arrive at boundary flip-flops in incorrect clock cycles. To guarantee that the number of clock cycles along any path does not change after optimization, we introduce relative timing references.

The concept of relative timing references can be explained in Fig. 3, where the boundary flip-flops include F1 and F4 and the flip-flops F2 and F3 in the middle are removed initially for optimization. To guarantee the correct function of F4, the latest arrival time s_z for stable signals at the input of F4 should be t_{su} before the rising clock edge at F4 and the earliest arrival time s'_z should be larger than t_h to make F4 latch data correctly. These constraints can be expressed as follows

$$s_z + t_{su} \leq T \tag{1}$$

$$s'_z \geq t_h. \tag{2}$$

The clock period T in (1) indicates that the signal should arrive at F4 within one clock period. Accordingly, (1)-(2) are defined according to the reference of the rising clock edge at the original F3. Similarly, the timing constraint at F3 in the original circuit is defined according to the reference of the rising clock edge at F2. At the boundary flip-flops, this definition still holds. The locations of the removed flip-flops such as F2 and F3 are called ***anchor points***. After removing all sequential components from the circuit under optimization,

these anchor points still provide relative timing information for boundary flip-flops. In VirtualSync, T is subtracted from the arrival time of signal when the signal passes an anchor point. When a signal finally arrives at a boundary flip-flop along a combinational path, its arrival time must have been converted by subtracting the number of flip-flops multiplied by T on the same path in the original circuit, so that (1)-(2) still hold.

C. Synchronizing Logic Waves by Delay Units

After removing all sequential components from the circuit under optimization, signals that are so fast that they reach boundary flip-flops too early need to be slowed down with delay units. Those signals traveling along the critical paths require no additional delay. The relative timing references, provided by the anchor points, guarantee that the number of clock cycles along any path is maintained after optimization. The objective of the optimization is to find the locations of delay units to make the circuit work at a given clock period T while reducing the area cost. Since it is not straightforward to determine the locations of delay units, we formulate this task as an ILP problem and solve it with a heuristic method.

D. Iterative Relaxation in VirtualSync

To apply the timing model above, we introduce a framework to identify the locations of delay units in an iterative way. In each iteration, necessary locations of sequential delay units are refined gradually. In the first step of the framework, we identify the locations at which sequential delay units are indispensable by emulating the delay effects of sequential delay units as virtual gaps. To refine the locations obtained from the first step, we incorporate clock/data-to-q delays of sequential delay units in the model. The last step is to legalize the timing of sequential delay units. To reduce the area overhead, buffers are allowed to be replaced with sequential delay units. A detailed description of this framework can be found in [24].

III. Timing Camouflage against Counterfeiting

In the traditional timing paradigm, all combinational paths work within one clock period. Within this paradigm, designers only need to focus on logic design without having to worry about the timing interference between different clock stages. Consequently, the functionality of a circuit depends only on its structure, a netlist thus carries all necessary design information. For instance, Fig. 4(a) shows a part of a digital circuit with three flip-flops F1, F2 and F3. If attackers can identify the types of combinational logic gates and their connection with F1, F2 and F3 through reverse engineering, they can reconstruct the netlist and then process it using a standard IC design flow. They can even revise and optimize the circuit freely. Thereafter, the design can be manufactured illegally, posing a big threat to IC vendors since the cost to develop a modern IC is extremely high.

To improve circuit security against counterfeiting, we invalidate the assumption that the netlist carries all design information by introducing wave-pipelining paths [25]. For example, the flip-flop in the middle in Fig. 4(a) can be removed to construct the circuit in Fig. 4(b) [26]. There are now two logic waves propagating along the combinational path from F1 to F3 simultaneously. If the second wave does not affect the propagation of the first wave, the two waves are then latched by F3 correctly, so that the functionality of the circuit is still maintained.

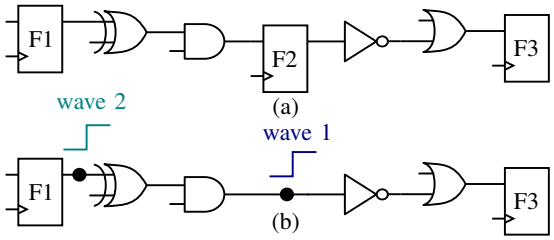

Fig. 4: Traditional timing and wave-pipelining. (a) Traditional clocking with single period. (b) Pipelining with two data waves.

Facing the camouflaged circuit with introduced wave-pipelining paths, attackers are forced to identify how many logic waves propagate simultaneously on the combinational paths between F1 and F3 even after the netlist is extracted through reverse engineering. If only one logic wave is assumed and the netlist is processed with a standard EDA flow, the reconstructed circuit does not function correctly because the data at the input of F3 is latched in incorrect clock cycles compared with the original circuit. To determine whether there are two data waves along combinational paths, timing information has to be extracted with additional effort. Accordingly, the functionality of the circuit depends on both its structure and the timing information of combinational paths.

A. Wave-Pipelining Construction

In constructing wave-pipelining paths to camouflage a circuit, the functionality of the circuit should be maintained. More precisely, the delay of wave-pipelining paths should be greater than T. Otherwise, data will be latched by flip-flops earlier than specified. The delays of these paths should also be less than $2T$ to guarantee that data is latched in correct clock cycles. The intuitive idea is to remove some flip-flops from the given circuit directly. For example, Fig. 5(a) illustrates this scenario, where ff_i is removed to construct wave-pipelining paths. However, the direct removal of flip-flops is not viable because on the left and right sides of ff_i, there are many short paths whose delays are very small. If we remove ff_i, these short paths will be connected together and their delays cannot satisfy the timing constraints of wave-pipelining paths.

To deal with the challenges from constructing wave-pipelining paths, we duplicate combinational logic in the original circuits, as shown in Fig. 5(b). We denote the flip-flop at which the wave-pipelining paths terminate as a WP flip-flop. To reduce the resource usage, on the right side of ff_i, we only duplicate combinational logic that drives WP flip-flops. Those combinational logic gates that do not drive any flip-flop in the original circuit are removed in Fig. 5(b). On the left side of ff_i, we firstly duplicate all the combinational logic to guarantee the same functionality with the original circuit. To reduce resource usage, we try to connect the input pins in the duplicated circuit with the original circuit. For example, if the input pin of the AND gate in the duplicated circuit in Fig. 5(b) is connected with that in the original circuit, the duplicated inverter can be removed. To satisfy the timing constraints of wave-pipelining paths, we apply gate sizing and buffer insertion to enlarge the

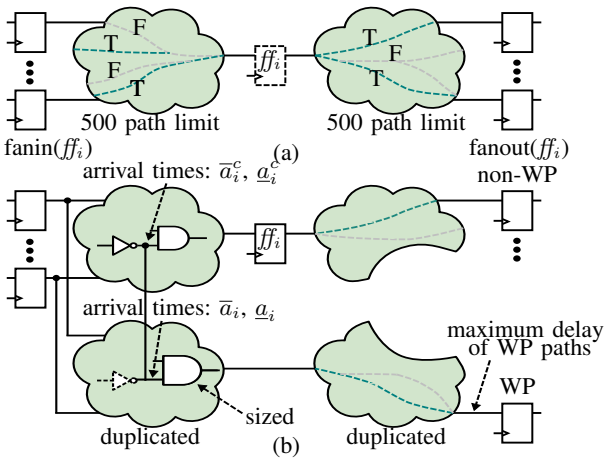

Fig. 5: Construction of WP path. (a) The flip-flop ff_i is removed directly. (b) Duplication of combinational logic and gate sizing.

delays of wave-pipelining paths. The task to construct wave-pipelining paths is formulated as an ILP problem and details can be found in [26].

B. Wave-Pipelining False Paths

Another challenge of the timing camouflage technique is that attackers could deploy exhaustive testing to determine whether the delays of combinational paths are greater than T. To prevent combinational paths from being tested, we introduce wave-pipelining false paths (WP false paths). The constructed WP false paths cannot be triggered by any test vectors. This concept can be explained using Fig. 6. After removing the flip-flop in the middle, the left and right path of it are connected to form a wave-pipelining path. If this wave-pipelining path is considered to work within one clock period, it is also a false path because signals switching at the beginning of this path cannot reach the final flip-flop. For example, if the signal v_2 has the value '1', it blocks signal propagation at the OR gate. Similarly, if the signal v_2 has the value '0', signal propagation is blocked at the AND gate. Consequently, attackers cannot test the delay of this path directly. It is also difficult to differentiate this path from an original false path in the circuit, since this path becomes a false path and about 75% of the combinational paths in a digital circuit are already false paths [27].

IV. Conclusion

In this paper, we have presented two techniques to break the confines of the traditional timing paradigm. With VirtualSync, circuit performance can be pushed even beyond the limit of the traditional timing paradigm. With timing camouflage, attackers face challenges from exhaustive testing and differentiation between wave-pipelining false paths and real false paths. These methods demonstrate that there is still a great potential in innovating the traditional timing paradigm for fast and secure circuits.

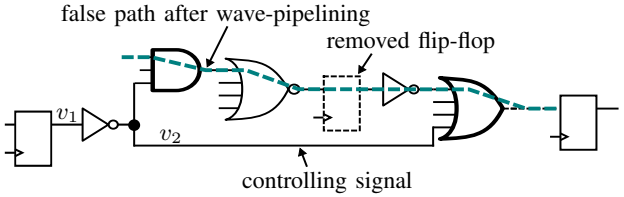

Fig. 6: A wave-pipelining false path formed with two true paths.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] Z. Lak and N. Nicolici, "A novel algorithmic approach to aid post-silicon delay measurement and clock tuning," *IEEE Trans. Comput.*, vol. 63, no. 5, pp. 1074–1084, 2014.

[6] 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.*, 2018.

[7] 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.

[8] 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.

[9] D. Blaauw, S. Kalaiselvan, K. Lai, W.-H. Ma, S. Pant, C. Tokunaga, S. Das, and D. M. Bull, "Razor II: In situ error detection and correction for PVT and SER tolerance," in *Proc. Int. Solid-State Circuits Conf.*, 2008, pp. 400–401.

[10] M. Fojtik, D. Fick, Y. Kim, N. R. Pinckney, D. M. Harris, D. Blaauw, and D. Sylvester, "Bubble razor: Eliminating timing margins in an ARM cortex-m3 processor in 45 nm CMOS using architecturally independent error detection and correction," *IEEE J. Solid-State Circuits*, vol. 48, no. 1, pp. 66–81, 2013.

[11] I. Kwon, S. Kim, D. Fick, M. Kim, Y.-P. Chen, and D. Sylvester, "Razor-lite: A light-weight register for error detection by observing virtual supply rails," *IEEE J. Solid-State Circuits*, vol. 49, no. 9, pp. 2054–2066, 2014.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "An analytical model for negative bias temperature instability," in *Proc. Int. Conf. Comput.-Aided Des.*, 2006, pp. 493–496.

[17] 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.

[18] 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.

[19] N. Koppaetzky, M. Metzdorf, R. Eilers, D. Helms, and W. Nebel, "RT level timing modeling for aging prediction," in *Proc. Design, Autom., and Test Europe Conf.*, 2016, pp. 297–300.

[20] 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.

[21] 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.

[22] 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.

[23] S. Karapetyan and U. Schlichtmann, "Integrating aging aware timing analysis into a commercial STA tool," in *Int. Symp. on VLSI Des., Aut. and Test (VLSI-DAT)*, 2015, pp. 1–4.

[24] 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.

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

[26] 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, pp. 91–96.

[27] 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.