VirtualSync: Timing Optimization by Synchronizing Logic Waves with Sequential and Combinational Components as Delay Units

Grace Li Zhang[1], Bing Li[1], Masanori Hashimoto[2], Ulf Schlichtmann[1]
[1]Chair of Electronic Design Automation, Technical University of Munich (TUM), Munich, Germany
[2]Department of Information Systems Engineering, Osaka University, Osaka, Japan
Email: {grace-li.zhang, b.li, ulf.schlichtmann}@tum.de, hasimoto@ist.osaka-u.ac.jp

Abstract

In digital circuit designs, sequential components such as flip-flops are used to synchronize signal propagations. Logic computations are aligned at and thus isolated by flip-flop stages. Although this fully synchronous style can reduce design efforts significantly, it may affect circuit performance negatively, because sequential components can only introduce delays into signal propagations instead of accelerating them. In this paper, we propose a new timing model, VirtualSync, in which signals, specially those along critical paths, are allowed to propagate through several sequential stages without flip-flops. Timing constraints are still satisfied at the boundary of the optimized circuit to maintain a consistent interface with existing designs. By removing clock-to-q delays and setup time requirements of flip-flops on critical paths, the performance of a circuit can be pushed even beyond the limit of traditional sequential designs. Experimental results demonstrate that circuit performance can be improved by up to 11.5% (average 3.1%) compared with that after thorough sizing and retiming, while the increase of area is still negligible.

1 Introduction

In digital circuit designs, clock frequency determines the timing performance of circuits. In the traditional timing paradigm, sequential components, e.g., edge-triggered flip-flops, synchronize signal propagations between pairs of flip-flops. Consequently, these propagations are blocked at flip-flops until a clock edge arrives. At an active clock edge, the data at the inputs of flip-flops are transferred to their outputs to drive the logic at the next stage. Therefore, combinational logic blocks are isolated by flip-flop stages. This fully synchronous style can reduce design efforts significantly, since only timing constraints local to pairs of flip-flops need to be met.

Within the traditional timing paradigm, many methods have been proposed to improve circuit performance. A widely adopted method is sizing, in which gates are sized to improve objectives such as clock frequency and area efficiency, while timing constraints between flip-flops are satisfied. For example, [1] introduces a fast and exact algorithm for simultaneous gate and wire sizing to minimize total area and propagation delay inside a circuit. In [2], a Lagrangian Relaxation (LR) based formulation together with a graph model is proposed to optimize timing slacks and power consumption simultaneously. In [3], a metaheuristic approach is developed to size logic gates that have the greatest impact on power-performance tradeoffs. This method guarantees slack, capacitance and slew constraints throughout the optimization process.

The second method to improve circuit performance in the traditional paradigm is retiming, which moves sequential components, e.g., flip-flops, but still preserves the correct functional behavior of circuits. In [4], an efficient algorithm is proposed to retime sequential circuits under both setup and hold constraints. The work in [5] demonstrates a maximum-flow-based approach to minimize the number of flip-flops. In [6], a new retiming method with a network-simplex algorithm is introduced for two-phase latch-based resilient circuits to reduce the overhead of normal and error detecting latches. Retiming for FPGA has been investigated in [7] to meet architecture constraints such as avoiding flip-flops through carry chains to guarantee a correct circuit function. A retimed circuit can be further improved by introducing intentional clock skews or latches to balance the delays of sequential stages with a finer granularity[8–12].

Wave-pipelining is the third method to improve circuit performance, where logic waves are allowed to propagate through combinational paths without intermediate sequential components. This method provides a mechanism to make the clock frequency of a circuit independent of the largest path delay, which limits circuit performance in traditional circuit designs [13]. As early as in [14], a linear method to minimize the clock period using wave pipelining is proposed. Recently, this method is also explored for majority-based beyond-CMOS technologies to improve the throughput of majority inverter graph (MIG) designs in [15].

The first two methods above can be used separately or jointly to improve circuit performance. However, sequential components are assumed to synchronize signal propagations in these methods, where no signal propagation through sequential components is allowed except at the clock edges. This synchronization with sequential components achieves many benefits such as reducing design efforts. However, it limits circuit performance in two regards. Firstly, sequential components have inherent clock-to-q delays and impose setup time. The former becomes a part of combinational paths driven by the corresponding flip-flops and the latter deprives a further part of the timing budget for the critical paths. Secondly, delay imbalances between flip-flop stages cannot be exploited since signal propagations are blocked at flip-flops instead of being allowed to propagate through flip-flops. Although clock skew scheduling can relieve this problem to some degree, it still suffers the inherent clock-to-q delays and setup time constraints of flip-flops. The third method above, wave-pipelining, allows signals to pass through sequential stages without flip-flops. However, this technique is not compatible with the traditional timing paradigm.

In this paper, we propose a new timing model, VirtualSync, which breaks the confines of the traditional timing paradigm. Our contributions are as follows:

- In the proposed new timing model, sequential components and combinational logic gates are both considered as delay units. Combinational logic gates add linear delays of the same amount to short and long paths, where sequential components provide non-linear delay effects, which provide different delay effects to fast and slow signal propagations.
- With the new timing model, a timing optimization framework is proposed to allocate sequential components only at necessary locations in the circuit to synchronize signal propagations, while the functionality of circuits is maintained. The absence of flip-flops at some sequential stages allows a virtual synchronization to provide identical functionality as in the original circuit. Consequently, the original clock-to-q delays and setup requirements along the critical paths can be removed to achieve a better circuit performance even beyond the limit of traditional sequential designs.
- Experimental results demonstrate that timing performance of circuits with the proposed framework can be improved by up to 11.5% (average 3.1%) compared with circuits after thorough sizing and retiming, with only a negligible increase of area.

DAC '18, June 24–29, 2018, San Francisco, CA, USA

ACM ISBN 978-1-4503-5700-5/18/06…$15.00
https://doi.org/10.1145/3195970.3196135

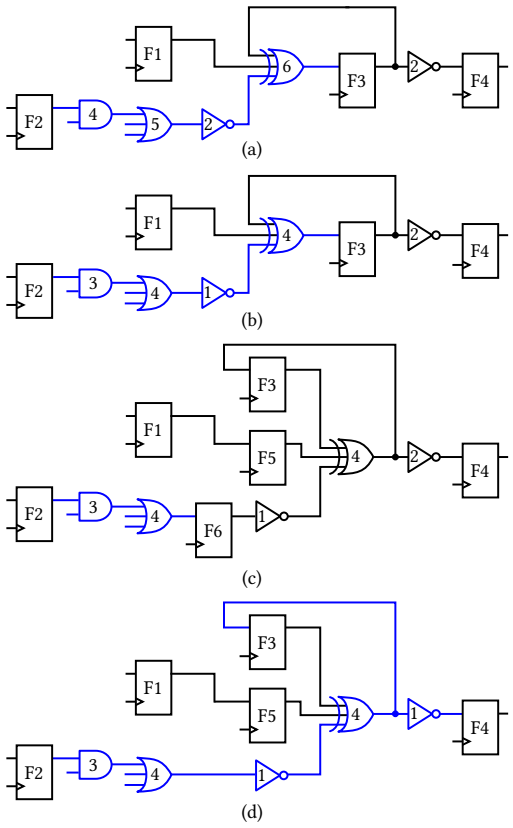

Figure 1: Timing optimization methods. Delays of logic gates are shown on the gates. The clock-to-q delay (t_{cq}), setup time (t_{su}) and hold time (t_h) of a flip-flop are 3, 1 and 1, respectively. (a) Original circuit. (b) Sized circuit. (c) Circuit after retiming. (d) Circuit after optimization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

The rest of this paper is organized as follows. In Section 2, we explain the motivation and the basic idea of the proposed method. The timing optimization problem is formulated in Section 3. In Section 4, we provide a detailed description of the proposed timing model. We describe implementation details of the proposed framework in Section 5. Experimental results are reported in Section 6. Conclusions are drawn in Section 7.

2 Background and Motivation

In traditional digital circuits, sequential components such as flip-flops synchronize signal propagations between pairs of flip-flops using a global clock signal, as shown in Fig. 1(a). The combinational path between F2 and F3 is critical with a path delay equal to 17. Assume that the clock-to-q delay, the setup time and the hold time of a flip-flop are 3, 1, and 1, respectively. The minimum clock period of this circuit is thus equal to 21.

To reduce the clock period, logic gates with smaller delays can be selected from the library to accelerate signal propagations on the critical paths of the circuit, at the cost of additional area overhead, leading to the circuit shown in Fig. 1(b), where the logic gates that are not on the critical path still have their original delays for the sake of saving area. After sizing, the minimum clock period of this circuit is reduced to 16 units. To reduce the clock period further, retiming can be deployed to move F3 to the left of the XOR gate as shown in Fig. 1(c), leading to a minimum clock period equal to 11.

The circuit in Fig. 1(c) has reached the limit of timing performance in the traditional timing model, and no other method except a logic redesign can reduce the clock period further. However, this strict timing constraint can still be relaxed by removing F6 from the circuit, leading to the circuit in Fig. 1(d). If the signal from from F2 can reach the sink flip-flops F3 and F4 after the next rising clock edge and before the rising edge two periods later, data can still be latched by F3 and F4 correctly. Since the inverter before F4 can also be sized further, the largest path delay is 16, which imposes a lower bound for the clock period as (16+1)/2=8.5, 22.7% lower than retiming.

Since F6 can be removed from the circuit without affecting its function in fact, it makes no contribution to the logic function or timing performance in Fig. 1(c). However, the flip-flop F5 in Fig. 1(c) cannot be removed, because the signal from F1 should also arrive at F4 later than one clock period. Without F5, the signal from F1 arrives at F4 even before the next rising clock edge, a loss of logic synchronization arises compared with the circuit in Fig. 1(a). Comparing Fig. 1(b) and Fig. 1(d), we can see that F3 in Fig. 1(b) simply blocks the fast path from F1 to F4 to avoid loss of logic synchronization or timing violations at F4, but it degrades the circuit performance by delaying the signal from F2 to F4 too.

The concept to allow logic signals to span several sequential stages without a flip-flop separating them is called *wave-pipelining* [13]. Previously, this technique has only been explored in the context of circuit design, where the numbers of waves on logic paths should be defined and their synchronization should be maintained by designers during the design phase. Since logic design and timing cannot be handled separately as in traditional synchronous designs, wave-pipelining becomes incompatible with the traditional fully synchronous design paradigm and prevents its adoption in practical designs. In VirtualSync, we introduce a new timing model that allows multiple waves on logic paths as a technique of timing optimization for circuits in the traditional design style. The resulting circuits still provide correct timing interfaces to sequential components, e.g., flip-flops, at the boundary of the optimized circuits to maintain timing compatibility.

3 Problem Formulation

In digital circuits, the essential function of sequential components is to delay signals along fast paths in a circuit. For example, in Fig. 1(d), F5 must be kept in the circuit to delay the signal propagation from F1 to F4. The sequential components that only sit on the critical path can thus be removed to improve circuit performance, such as F6 in Fig. 1(d).

In the VirtualSync framework, we remove all flip-flops and then identify the necessary locations to block fast signals using combinational gates and sequential components, e.g., buffers, flip-flops, and latches. The advantage of this formulation is that it is possible to insert the minimum number of delay units into the circuit to achieve the theoretical minimum clock period.

The problem formulation of VirtualSync is described as follows:
Given: the netlist of a digital circuit; the delay information of the circuit; the target clock period T.
Output: a circuit with adjusted number and locations of sequential components; logic gates with new sizes; inserted delay units, e.g., buffers.
Objectives: the circuit should maintain the same function viewed from the sequential components at the boundary of the optimized circuit; the target timing specification should be met; the area of the optimized circuit should be reduced.

4 VirtualSync Timing Model

4.1 Delay Units

In the VirtualSync framework, we first remove all sequential components, flip-flops, from the circuit under optimization. Consequently, logic synchronization may be lost because signals across fast paths may arrive at flip-flops in incorrect clock cycles, e.g., earlier than specified, or timing violations may be incurred. In addition, signals along combinational loops should also be blocked to avoid the loss of logic synchronization. For example, in Fig. 1(d), the combinational loop across the XOR gate must have a sequential component; otherwise a signal loses synchronization after traveling across it many times.

To slow down a signal, three different components can be used as delay units, namely, combinational gates such as buffers, flip-flops, and latches, which exhibit different delay characteristics, as shown in Fig. 2, where the term ***input gap*** refers to the difference of arrival times of two signals at a delay unit, and the term ***output gap*** represents the difference between their arrival times after they pass through the unit.

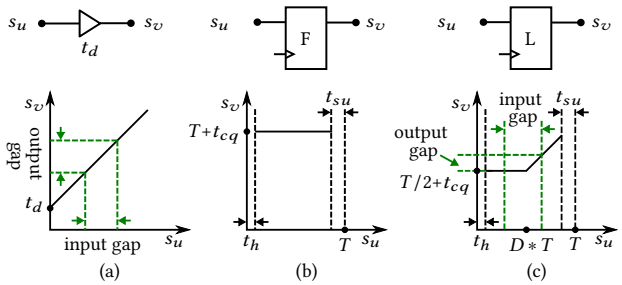

Figure 2: Properties of delay units. (a) Linear delaying effect of a combinational delay unit. (b) Constant delaying effect of a flip-flop. (c) Piecewise delaying effect of a latch.

In Fig. 2(a), a combinational delay unit adds the same amount of delay to any input signal. Consequently, the arrival time s_v at the output of the combinational delay unit is linear to the arrival time s_u at the input of the delay unit. Therefore, the absolute gap between arrival times of signals through short and long paths does not change when a combinational delay unit is passed through.

In delaying input signals, a flip-flop, as a sequential delay unit, behaves completely differently from a combinational delay unit, as shown in Fig. 2(b). If the arrival time of a signal falls into the time window $[t_h, T - t_{su}]$, where t_h is the hold time and t_{su} is the setup time, the output signal always leaves at the time $T + t_{cq}$, with t_{cq} as the clock-to-q delay of the flip-flop. Therefore, the gap between the arrival times of two signals reaching the input of a flip-flop is always reduced to zero at the output of the flip-flop. This is a very useful property because the delays of short paths and long paths in a circuit may differ significantly after all sequential components are removed from the circuit under optimization. For many short paths, it is not possible to increase their delays by adding combinational delay units such as buffers to them, because the combinational delay units on the short paths may also appear on other long paths. The increased delays along long paths might affect circuit performance negatively. Flip-flops are thus of great use in this scenario, because short paths receive more delay padding than long paths to align logic waves in the circuit.

As the second type of sequential delay units, level-sensitive latches have a delay property combining those of combinational delay units and flip-flops, as shown in Fig. 2(c), where $0 < D < 1$ is the duty cycle of the clock signal. Assume that a latch is non-transparent in the first part of the clock period and transparent in the second part of the clock period. If two input signals arrive at a latch when it is non-transparent, the output gap is reduced to zero. If both signals arrive at a latch when it is transparent, the gap remains unchanged. However, if the fast signal reaches the latch when it is non-transparent while the slow signal reaches it when it is transparent, the output gap of the two signals is neither zero nor unchanged. Instead, it takes a value between the two extreme cases as illustrated in Fig. 2(c). This property gives us more flexibility to modulate signals with different arrival times, specifically those along critical paths where fast signals require more delay padding and slow signals should not be affected.

4.2 Relative Timing References

In Fig. 1(a), if all the logic gates and flip-flop F3 are considered as the the circuit under optimization, F1, F2 and F4 are thus the boundary flip-flops. No matter how signals inside the circuit propagate, the function of the whole circuit is still maintained if we can guarantee that for any input pattern at flip-flops F1 and F2 the circuit produces the same result at F4 at the same clock cycle as the original circuit.

Consider a general case in Fig. 3, where F1 and F4 are the boundary flip-flops and F2 and F3 are removed in the initial circuit for optimization. At F4, the arrival times are required to meet the setup and hold time constraints, written as

$$s_z + t_{su} \le T \quad (1)$$

$$s'_z \ge t_h \quad (2)$$

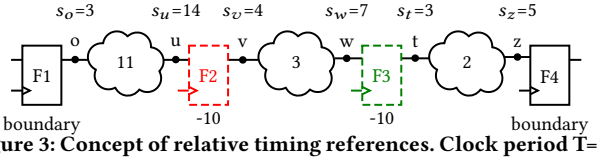

Figure 3: Concept of relative timing references. Clock period T=10. Clock-to-q delay t_{cq}=3. Both setup time t_{su} and hold time t_h are equal to 1. F3 is kept in the optimized circuit and F2 is not included.

where s_z and s'_z are the latest and earliest arrival times at z. These two constraints in fact are defined with respect to the rising clock edge at F3, since the clock period T in (1) shows that the signal should arrive at F4 within one clock period. Although F2 and F3 are removed from the circuit, the constraints at F4 should still be the same as (1)-(2) to maintain the compatibility of the timing interface at the boundary flip-flops.

In the general case in Fig. 3, we can also observe that the timing constraint at F3 in the original circuit is also defined with respect to the rising clock edge at F2. This definition can be chained further back until the source flip-flop F1 at the boundary is be reached. We call the locations of these removed flip-flops such as F2 and F3 ***anchor points***. After all sequential components are removed from the circuit under optimization, these anchor points still allow to relate timing information to boundary flip-flops. Every time when a signal passes an anchor point, its arrival time is converted by subtracting T in VirtualSync. When a signal finally arrives at a boundary flip-flop along a combinational path, its arrival time must be converted so many times as the number of flip-flops on the path, so that (1)-(2) is still valid.

In Fig. 3, assume that F2 is removed but F3 is inserted back in the optimized circuit. The arrival time s_u is subtracted by the clock period T=10 to convert it with respect to the time at F1, leading to s_v=4. The arrival time s_w is defined with respect to the previous flip-flop before F3, so that the timing constraints can be checked using (1)-(2). Since the arrival time before F4 should meet its timing constraints, F3 thus cannot be removed. Otherwise, the arrival time s_t would be equal to 7-10=-3. Accordingly, the arrival time s_z becomes -3+2=-1, definitely violating the hold time constraint in (2).

Since F3 is kept in the optimized circuit, it introduces the delay with the property shown in Fig. 2(b). The arrival time after this sequential delay unit thus becomes $T + t_{cq}$=13. This signal at t in Fig. 3 also passes an anchor point. Therefore, the arrival time s_t is equal to 3, leading to no timing violation at F4. This example demonstrates that the timing constraints at the boundary flip-flops force the usage of the internal sequential delay units. The model to insert these delay units automatically will be explained in the next section.

4.3 Synchronizing Logic Waves by Delay Units

With all flip-flops removed from the circuit under optimization, we only need to delay signals that are so fast that they reach boundary flip-flops too early; signals that propagate slowly are already on the critical paths, thus requiring no additional delay. Since it is not straightforward to determine the locations for inserting additional delays, we formulate this task as an ILP problem and solve it later with introduced heuristic steps. The values of variables in the following sections are determined by the solver, unless they are declared as constants explicitly.

The scenario of delay insertion at a circuit node, i.e., a logic gate, is illustrated in Fig. 4, where a combinational delay unit ξ_{uv} may be inserted, the original delay of the logic gate may be sized, and a sequential delay unit may be inserted to block fast and slow signals with different delays. Furthermore, the number of flip-flops between w and t in the original circuit is represented by an integer constant λ_{tz}. When $\lambda_{tz} \ge 1$, an anchor point is found at the location between t and z. λ_{tz} is used to convert arrival times.

4.3.1 Combinational delay unit and gate sizing

In Fig. 4, the delay at the circuit node can be changed by sizing the delay of the logic gate, e.g., the XOR gate in Fig. 4. For the case that the required gate delay exceeds the largest permissible value, a combinational delay unit is inserted at the corresponding input. For convenience, we assume

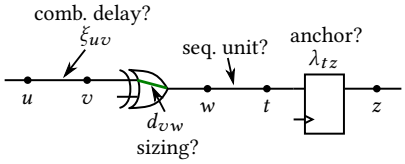

Figure 4: Delay insertion model in VirtualSync.

the combinational delay unit inserted at the input is implemented with buffers. The relation between the arrival times u and w is thus expressed as

$$s_w \geq s_u + \xi_{uv} * r^u + d_{vw} * r^u \tag{3}$$

$$s'_w \leq s'_u + \xi_{uv} * r^l + d_{vw} * r^l \tag{4}$$

where s_u, s'_u, s_w and s'_w are the latest and earliest arrival times of node u and w , respectively. ξ_{uv} is the extra delay introduced by an inserted buffer and d_{vw} is the pin-to-pin delay of the logic gate. If ξ_{uv} is reduced to 0 after optimization, no buffer is required in the optimized circuit. The $\leq$ and $\geq$ relaxations of the relation between arrival times guarantee that only the latest and the earliest arrival times from multiple inputs are propagated further. r^u and r^l are two constants to reserve a guard band for process variations, so that $r^u > 1$ and $r^l < 1$.

4.3.2 Insertion of sequential delay units

Since arrival times through long and short paths reaching w may have a large difference, we may need to insert sequential delay units to delay the fast signal more than the slow signal. This can be implemented with the sequential units shown in Fig. 2, where the gap between the arrival times is reduced after passing a sequential delay unit, either a flip-flop or a latch. To insert a sequential delay unit, three cases need to be examined.

Case 1: No sequential delay unit is inserted between w and t in Fig. 4, so that

$$s_t \geq s_w \tag{5}$$

$$s'_t \leq s'_w. \tag{6}$$

Case2: A flip-flop is inserted between w and t. Assume the flip-flop works at a rising clock edge. As shown in Fig. 2(b), a flip-flop only works properly in a region t_h after the rising clock edge and t_{su} before the next rising clock edge. Therefore, we need to bound the arrival times s_w and s'_w into such a region by

$$s_w, s'_w \geq N_{wt} * T + \phi_{wt} + t_h * r^u \tag{7}$$

$$s_w, s'_w \leq (N_{wt} + 1) * T + \phi_{wt} - t_{su} * r^u \tag{8}$$

where N_{wt} is an integer variable determined by the solver. T is the given clock period. ϕ_{wt} is phase shift of the clock signal. The available values of ϕ_{wt} can be set by designers. If only one clock signal is available, ϕ_{wt} can be set to 0 and T/2 to emulate flip-flops working at rising and falling clock edges.

When the input arrival times fall into the valid region of a flip-flop as constrained by (7)–(8), the signal always starts to propagate from the next active clock edge, so that the constraints can be written as

$$s_t \geq (N_{wt} + 1)T + \phi_{wt} + t_{cq} * r^u \tag{9}$$

$$s'_t \leq (N_{wt} + 1)T + \phi_{wt} + t_{cq} * r^l. \tag{10}$$

Case3: A level-sensitive latch is inserted between w and t. To be consistent with the active region of flip-flops, we assume that the latches are transparent when the clock signal is equal to 0. We can then bound the arrival times at w the same as (7)–(8).

As illustrated in Fig. 2(c), the latch is non-transparent in the first part of the region and transparent in the second region. Accordingly, the latest time a signal leaves the latch can be expressed as

$$s_t \geq N_{wt} * T + \phi_{wt} + D * T + t_{cq} * r^u \tag{11}$$

$$s_t \geq s_w + t_{dq} * r^u \tag{12}$$

where (11) corresponds to the case that the latch is non-transparent, so that the signal leaves the latch at the moment the clock switches to 1. D is the duty cycle of the clock signal with $0 < D < 1$. (12) corresponds to the case that the latch is transparent, so that only the delay of the latch is added to s_w. t_{dq} is the data-to-q delay of the latch.

The earliest time a signal leaves the latch is, however, imposed by a constraint in the less-than-max form as in [16],

$$s'_t \leq \max\{N_{wt} * T + \phi_{wt} + D * T + t_{cq} * r^l,\ \ s'_w + t_{dq} * r^l\} \tag{13}$$

which cannot be linearized easily. In the VirtualSync framework, the purpose of introducing the sequential delay unit is to delay the short path as much as possible. This effect happens when a signal arrives at a non-transparent latch. Therefore, we impose the arrival times of fast signals to be positioned in the non-transparent region, expressed as

$$N_{wt} * T + \phi_{wt} + t_h * r^u \leq s'_w \leq N_{wt} * T + \phi_{wt} + D * T \tag{14}$$

while relaxing (13) as

$$s'_t \leq N_{wt} * T + \phi_{wt} + D * T + t_{cq} * r^l. \tag{15}$$

When inserting the sequential delay unit, each of the three cases above can happen in the optimized circuit. We use an integer variable to represent the selection and let the solver determine which case happens during the optimization.

4.3.3 Reference shifting with respect to anchor points

The arrival times in the model need to be converted each time when an anchor point is passed. The constant λ_{tz} represents the number of flip-flops at such a point in the original circuit. In Fig. 4, the arrival time at z is shifted as

$$s_z = s_t - \lambda_{tz} T. \tag{16}$$

4.3.4 Wave non-interference condition

Since we allow multiple waves to propagate along a combinational path, we need to guarantee that the signal of the next wave starting from a boundary flip-flop never catches the signal of the previous wave starting from the same flip-flop [13]. This constraint should be imposed to every node in the circuit. For example, the constraint for node u is written as

$$s_u + t_{stable} \leq s'_u + T \tag{17}$$

where t_{stable} is the minimum gap between two consecutive signals.

4.3.5 Overall formulation

The introduction of the relative timing references, or the anchor points, in Section 4.2 guarantees that the number of clock cycles along any path does not change after optimization. With the timing constraints (1)-(2) at boundary flip-flops, the correct function of the optimized circuit is always maintained, without requiring any change in other function blocks.

The constraints (1)–(17) excluding (13) need to be established at each node in the circuit after flip-flops are removed. The appearance of the combinational and sequential delay units needs to be determined by the solver. The delays of logic gates should also be sized. The objective of the optimization is to find a solution to make the circuit work at a given clock period T, while reducing the area cost. Taking all these factors into account, the straightforward ILP formulation may become insolvable. In practice, however, this technique only needs to be applied to isolated circuit parts containing critical paths. In addition, we introduce heuristic techniques to overcome this scalability problem, as explained in the following section.

5 Iterative Relaxation in VirtualSync

In applying the timing model above, we introduce a framework to identify the locations of delay units iteratively. The flow of this framework is shown in Fig. 5. The basic strategy is to remove flip-flops along critical paths to eliminate the inherent clock-to-q delays and setup time. Thereafter, fast signals of short paths will be blocked to guarantee the correct functionality by inserting the minimum number of sequential delay units and buffers. To reduce area overhead, buffers are replaced with sequential delay units.

5.1 Emulation of Sequential Delay Units

After we remove all the flip-flops from the original circuit, the short paths may have extremely small delays. The gap between these delays and those of long paths is very large. It cannot be reduced with combinational delay units since they introduce the same delays to the fast and slow signals as shown in Fig. 2(a). Instead, only sequential delay units are able to reduce this gap so that the fast and slow signals still arrive at boundary flip-flops within the same clock cycle as those of the original circuit.

In the first step of the framework, we identify the locations at which sequential delay units are indispensable. Without these units, the fast and slow signals, such as those within feedback loops, may not be aligned

properly into the correct clock cycles, even though unlimited combinational delay units can be inserted into the circuit. In practice, however, it is not easy to identify the exact locations of these units using the exact and complete model in (5)–(15) directly. To solve this problem, we first emulate the delay effects of sequential delay units shown in Fig. 2(b)–(c), in which sequential delay units provide different delay paddings for short and long paths. Therefore, we use two variables δ_{wt} and δ'_{wt} to emulate different delays to be padded to slow and fast signals, respectively. When signals travel from u to z in Fig. 4, the relation of arrival times from nodes u to z can be written as

$$s_z \geq s_u + \xi_{uv} * r^u + d_{vw} * r^u + \delta_{wt} - \lambda_{tz} T \tag{18}$$

$$s'_z \leq s'_u + \xi_{uv} * r^l + d_{vw} * r^l + \delta'_{wt} - \lambda_{tz} T \tag{19}$$

$$\delta_{wt} \leq \delta'_{wt} \tag{20}$$

$$s'_u + \delta'_{wt} \leq s_u + \delta_{wt} \tag{21}$$

where the variables δ_{wt} and δ'_{wt} emulate delays introduced by sequential delay units. (20) specifies that the fast signal should be padded with more delays than the slow signal. (21) specifies that the arrival time of the fast signal should not exceed the arrival time of the slow signal after padding.

The optimization problem to find the potential locations of sequential delay units is thus written as

$$\text{minimize} \quad \alpha \sum_G (\delta'_{wt} - \delta_{wt}) + \beta \sum_G (\delta'_{wt} + \xi_{uv}) - \gamma \sum_G d_{vw} \tag{22}$$

$$\text{subject to} \quad (17)\text{–}(21) \text{ for each gate in } G \tag{23}$$

$$\text{constraints (1)–(2) for each boundary flip-flop} \tag{24}$$

where G is the set of all logic gates in the original circuit. This optimization problem also maximizes the overall delays of logic gates in the circuit, so that not only the inserted delays but also the area of the circuit can be reduced. α, β and γ are constants, set to 100,10,10, to specify the balance between sequential delay units, inserted buffers and logic gates roughly. Solving the optimization problem above identifies nodes with unequal padding delays δ_{wt} and δ'_{wt}, indicating potential locations of sequential delay units, as a set S. These delays may still violate the exact constraints in (5)–(15), so that they need to be refined further.

5.2 Modeling with Clock/Data-to-Q Delays of Sequential Delay Units

The optimization problem (22)–(24) does not consider the inherent clock-to-q delays of flip-flops and data-to-q delays of latches. Since these delays are introduced only at locations where sequential delay units are inserted, they need to be modeled for all the locations S returned by the previous step. We introduce a binary variable x_{wt} to represent whether a sequential delay unit appears at a location from S, and revise the constraints (18)–(19) as

$$s_z \geq s_u + \xi_{uv} * r^u + d_{vw} * r^u + x_{wt}\delta_{wt} + x_{wt} t_{cd\to q} * r^u - \lambda_{tz} T \tag{25}$$

$$s'_z \leq s'_u + \xi_{uv} * r^l + d_{vw} * r^l + x_{wt}\delta'_{wt} + x_{wt} t_{cd\to q} * r^l - \lambda_{tz} T \tag{26}$$

where $t_{cd\to q}$ represents clock-to-q delay or data-to-q delay, approximated with the same value for simplicity, and the delays δ_{wt}, δ'_{wt} and $t_{cd\to q}$ are only valid when x_{wt} is equal to 1. The inclusion of the binary variables x_{wt} is very computation-intensive, so that they can only be dealt with after the potential locations of sequential delay units are reduced to S by solving (22)–(24). Since x_{wt} is a binary variable, the multiplications $x_{wt}\delta_{wt}$ and $x_{wt}\delta'_{wt}$ can be converted into equivalent linear forms so that the overall formulation is still an ILP problem.

Considering the inherent delays of sequential delay units, their locations can be refined further by solving the optimization problem as

$$\text{minimize} \quad \alpha \sum_{G/S} (\delta'_{wt} - \delta_{wt}) + \beta \sum_{G/S} (\delta'_{wt} + \xi_{uv}) - \gamma \sum_G d_{vw} \tag{27}$$

$$\text{subject to} \quad (17)\text{–}(21) \text{ for each gate in } G/S \tag{28}$$

$$(17), (20)\text{–}(21) \text{ and } (25)\text{–}(26) \text{ for each gate in } S \tag{29}$$

$$\text{constraints (1)–(2) for each boundary flip-flop.} \tag{30}$$

In the implementation, we also constrained the lower bound of $\delta'_{wt} - \delta_{wt}$ when x_{wt} is equal to 1 and lower it iteratively, so that the most important locations for inserting sequential delay units are identified first, as illustrated in Fig. 5. The iterations terminate when no different δ_{wt} and δ'_{wt} exist, indicating no indispensable sequential delay units are required to align fast and slow signals. The refined locations of sequential delay units from this step are returned as a set S_d.

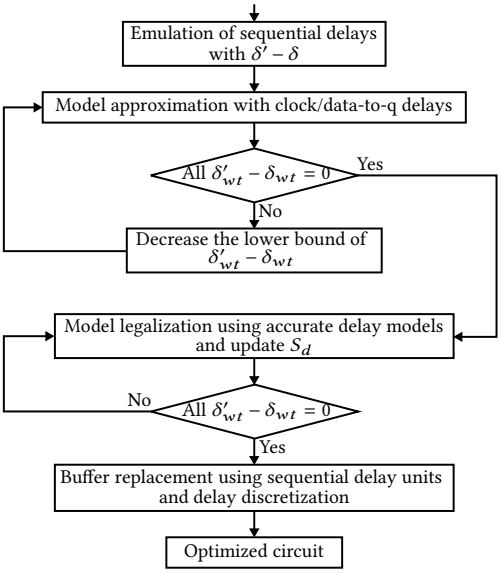

Figure 5: VirtualSync flow.

5.3 Model Legalization for Timing of Sequential Delay Units

In this step, the complete model described in Section 4.3 is applied to the locations in S_d to generate sequential delay units that are really required in the circuit. The optimization problem is described as

$$\text{minimize} \quad \alpha \sum_{G/S_d} (\delta'_{wt} - \delta_{wt}) + \beta \sum_{G/S_d} (\delta'_{wt} + \xi_{uv}) - \gamma \sum_G d_{vw} \tag{31}$$

$$\text{subject to} \quad (17)\text{–}(21) \text{ for each gate in } G/S_d \tag{32}$$

$$(5)\text{–}(12) \text{ and } (14)\text{–}(17) \text{ for each gate in } S_d \tag{33}$$

$$\text{constraints (1)–(2) for each boundary flip-flop.} \tag{34}$$

After solving the optimization above, there might still be different δ_{wt} and δ'_{wt} in G/S_d, because the timing legalization of sequential delay units with the complete model in Section 4.3 may invalidate some locations in S_d. The accurate sequential delay model is applied to these new locations iteratively, until no different δ_{wt} and δ'_{wt} exists, indicating the remaining timing synchronization can be achieved with buffers and gate sizing directly.

5.4 Buffer Replacement with Sequential Units

After solving (31)–(34), delays d_{vw} of logic gates are discretized according to the library. Buffer delays ξ_{uv} are also determined. If ξ_{uv} is large, several buffers are needed for its implementation. As shown in Fig. 2, sequential delay units can introduce a very large delay. For example, a flip-flop can introduce a delay as large as $T + t_{cq} - t_h$, if the incoming signal arrives at the flip-flop right after a clock edge. According to this observation, we iteratively replace buffers with large delays using sequential delay units to reduce area. In each iteration, the accurate sequential model (5)–(12) and (14)–(17) is applied to guarantee these new sequential delay units are valid. The iteration stops when no buffer can be replaced by sequential units. Buffers that cannot be replaced by sequential delay units are implemented directly in the optimized circuit.

6 Experimental Results

The proposed method was implemented in C++ and tested using a 3.20 GHz CPU. We demonstrate the results using circuits from the ISCAS89 benchmark set and the TAU 2013 variation-aware timing analysis contest as shown in Table 1. The number of flip-flops and the number of logic gates are shown in the columns $\boldsymbol{n_s}$ and $\boldsymbol{n_g}$, respectively. The benchmark circuits were sized using a 45 nm library. To tolerate process variations, 10% of timing margin was assigned, so that r^u and r^l in previous sections were

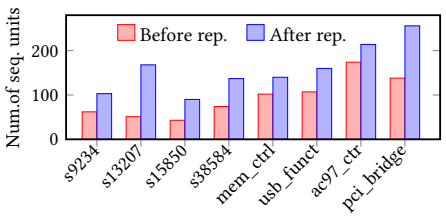

Figure 6: Comparison of sequential delay units after buffer replacement.

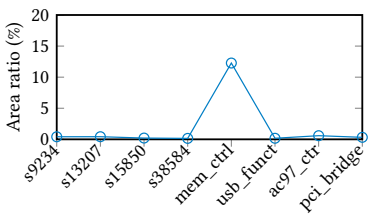

Figure 7: Area comparison before and after buffer replacement.

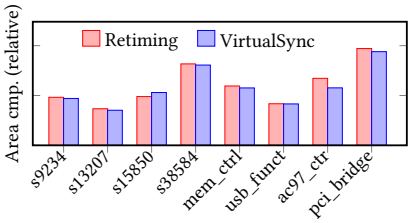

Figure 8: Area comparisons with retiming&sizing with the same clock period.

Table 1: Results of VirtualSync

Circuit			Cri. Part		Opt. Circuit			Comparison		Runtime
	n_s	n_g	n_{cs}	n_{cg}	n_f	n_l	n_b	n_t	n_a	$t(s)$
s5378	179	2779	35	1877	11	14	94	11.5%	2.84%	121.6
s9234	228	5597	91	3981	58	45	91	2.5%	-5.17%	7251.1
s13207	669	7951	191	3483	95	73	52	2.5%	-1.09%	3121.6
s15850	534	9772	71	3847	72	18	26	0%	6.01%	289.97
s38584	1452	19253	126	9498	62	75	46	0.5%	-0.50%	1142.3
systemcdes	190	3266	92	3232	90	81	227	3.5%	2.43%	7310.5
mem_ctrl	1065	10327	136	7500	101	39	140	3.5%	0.97%	3750.1
usb_funct	1746	14381	138	5378	123	37	60	4%	0.21%	1211.7
ac97_ctrl	2199	9208	237	4873	42	172	218	0%	-9.76%	2936.8
pci_bridge	3321	12494	239	9510	188	68	338	3%	0.05%	7418.5

set to 1.1 and 0.9, respectively. The allowed phase shifts ϕ_{wt} in previous sections are 0, T/4, T/2 and 3T/4. The ILP solver used in the VirtualSync framework was Gurobi.

For timing optimization, combinational paths whose delays were within 95% of the largest path delay in the circuit were selected. The source and sink flip-flops of these paths were allowed to be removed, while the other flip-flops in the circuits were considered as boundary flip-flops. All the combinational logic gates that can reach the flip-flops at the sources or sinks of these selected paths through a combinational path in the original circuit are considered as the critical parts of a circuit together. This extraction of the critical parts of a circuit in fact allows wave-pipelining within three sequential stages. In real circuits, the extracted critical parts may overlap, thus allowing wave-pipelining with more than three stages. Since the original circuits have been sized to reduce the clock period, the critical parts of the circuits still occupied a large portion of original circuits. As shown in the n_{cs} and n_{cg} columns in Table 1, more than 7% of flip-flops and more than 35% of logic gates have been selected for timing optimization.

The column n_f and n_l show the numbers of flip-flops and latches after optimization, respectively. The sums of these numbers are comparable or even smaller than the numbers of flip-flops in the original critical parts of the circuits. The numbers of extra inserted buffers to match arrival times are shown in the column n_b. Compared with the number of original logic gates shown in the column n_{cg}, these numbers show that the cost due to the inserted buffers is still acceptable.

To verify the improvement of circuit performance, we gradually reduced the clock period by 0.5% of the clock period obtained from combining retiming and sizing. The column n_t in Table 1 shows the clock period reduction compared with the circuits after retiming&sizing. The maximum and average reduction are 11.5% and 3.1%, respectively, which resulted from the compensations between several flip-flop stages and the removal of clock-to-q delays and setup time requirements on critical paths. For most cases, the minimum clock periods have been pushed even further than those from retiming&sizing, the limit of the traditional sequential design. This comparison demonstrates that the proposed method can potentially improve circuit performance specially at a late stage of timing closure further, because no circuit redesign is required. The area increase compared with retiming&sizing is shown in column n_a. In the cases with area increase, the overhead is still negligible; in other cases, the area is even smaller because unnecessary flip-flops were removed in the proposed framework, whereas in retiming flip-flops can only be moved instead of being removed. The last column t_r in Table 1 shows the runtime of the proposed method. Since the ILP formulation with the complete model in Section 4.3 is NP-hard, it is impractical to find a solution with respect to area and clock period. In the experiments, the runtime with iterative relaxations is acceptable at a late stage of timing closure, but still has space for further improvement.

In the proposed framework, sequential delay units are first inserted only at necessary locations to delay signal propagations. Afterwards, more of them are used to replace buffers to reduce area, as described in Section 5. Figure 6 shows the numbers of sequential delay units before and after buffer replacement, which shows a clear increase of the number of such delay units to replace buffers. Figure 7 shows the area comparison after this replacement. In most test cases, the area taken by the sequential delay units and buffers in the optimized circuits is less than 1% of those replaced buffers, demonstrating the efficiency of sequential delay units in delaying fast signals.

The comparison of the area overhead in Table 1 is between the method retiming&sizing with its own clock period and the proposed method with a smaller clock period. To demonstrate the area efficiency of the proposed method, we also compared the proposed method and the retiming&sizing with the same clock period from the latter. The results are shown in Fig. 8. In most cases, the area overhead with our framework is smaller.

7 Conclusion

In this paper, we have proposed a new timing model, VirtualSync, in which sequential components and combinational logic gates are considered as delay units. They provide different delay effects on signal propagations on short and long paths. With this new timing model, a timing optimization framework has been proposed to insert delay units only at necessary locations. With this technique, circuit performance can be improved by up to 11.5% with a negligible increase of area overhead. Future work includes exploring more efficient methods, e.g., Lagrangian Relaxation, to reduce the runtime.

References

[1] C.-P. Chen, C. C. Chu, and D. Wong, "Fast and exact simultaneous gate and wire sizing by lagrangian relaxation," in *Proc. Design Autom. Conf.*, 1998, pp. 617–624.

[2] M. M. Ozdal, S. Burns, and J. Hu, "Gate sizing and device technology selection algorithms for high-performance industrial designs," in *Proc. Int. Conf. Comput.-Aided Des.*, 2011, pp. 724–731.

[3] J. Hu, A. B. Kahng, S. Kang, M.-C. Kim, and I. L. Markov, "Sensitivity-guided metaheuristics for accurate discrete gate sizing," in *Proc. Int. Conf. Comput.-Aided Des.*, 2012, pp. 233–239.

[4] C. Lin and H. Zhou, "An efficient retiming algorithm under setup and hold constraints," in *Proc. Design Autom. Conf.*, 2006, pp. 945–950.

[5] A. P. Hurst, A. Mishchenko, and R. K. Brayton, "Scalable min-register retiming under timing and initializability constraints," in *Proc. Design Autom. Conf.*, 2008, pp. 534–539.

[6] H.-L. Wang, M. Zhang, and P. A. Beerel, "Retiming of two-phase latch-based resilient circuits," in *Proc. Design Autom. Conf.*, 2017, pp. 1–6.

[7] D. R. Singh, V. Manohararajah, and S. D. Brown, "Incremental retiming for FPGA physical synthesis," in *Proc. Design Autom. Conf.*, 2005, pp. 433–438.

[8] 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.

[9] ——, "EffiTest: Efficient delay test and statistical prediction for configuring post-silicon tunable buffers," in *Proc. Design Autom. Conf.*, 2016, pp. 60:1–60:6.

[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] 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.

[12] G. L. Zhang, B. Li, Y. Shi, H. Jiang, and U. Schlichtmann, "EffiTest2: Efficient delay test and prediction for post-silicon clock skew configuration under process variations," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, 2018, doi: 10.1109/TCAD.2018.2818713.

[13] 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, Sep. 1998.

[14] M. J. C. D. A. Joy, "Clock period minimization with wave pipelining," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 12, no. 4, pp. 461–472, Apr. 1993.

[15] O. Zografos, A. D. Meester, E. Testa, M. Soeken, P. E. Gaillardon, G. D. Micheli, L. Amarù, P. Raghavan, F. Catthoor, and R. Lauwereins, "Wave pipelining for majority-based beyond-CMOS technologies," in *Proc. Design, Autom., and Test Europe Conf.*, 2017, pp. 1306–1311.

[16] K. Sakallah, T. Mudge, and O. Olukotun, "checkT_c and minT_c: Timing verification and optimal clocking of synchronous digital circuits," in *Proc. Int. Conf. Comput.-Aided Des.*, 1990, pp. 552–555.