POLITECNICO DI TORINO
Repository ISTITUZIONALE

Mix & Latch: An Optimization Flow for High-Performance Designs with Single-Clock Mixed-Polarity Latches and Flip-Flops

Original

Mix & Latch: An Optimization Flow for High-Performance Designs with Single-Clock Mixed-Polarity Latches and Flip-Flops / Minnella, Filippo; Cortadella, Jordi; Casu, Mario R.; Lazarescu, Mihai T.; Lavagno, Luciano. - In: IEEE ACCESS. - ISSN 2169-3536. - ELETTRONICO. - 11:(2023), pp. 1-1. [10.1109/ACCESS.2023.3265809]

Availability:
This version is available at: 11583/2977854 since: 2023-04-11T07:18:27Z

Publisher:
IEEE

Published
DOI:10.1109/ACCESS.2023.3265809

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

01 September 2025

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 4 November 2022, accepted 20 December 2022, date of publication 10 April 2023, date of current version 13 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3265809

RESEARCH ARTICLE

Mix & Latch: An Optimization Flow for High-Performance Designs With Single-Clock Mixed-Polarity Latches and Flip-Flops

**FILIPPO MINNELLA[1], JORDI CORTADELLA[2], (Fellow, IEEE),
MARIO R. CASU[1], (Senior Member, IEEE),
MIHAI T. LAZARESCU[1], (Senior Member, IEEE),
AND LUCIANO LAVAGNO[1], (Senior Member, IEEE)**
[1]Department of Electronic and Telecommunications, Politecnico di Torino, 10129 Turin, Italy
[2]Department of Computer Science, Universitat Politècnica de Catalunya, 08034 Barcelona, Spain

Corresponding author: Filippo Minnella (filippo.minnella@polito.it)

ABSTRACT Flip-flops are the most used sequential elements in synchronous circuits, but designs based on latches can operate at higher frequencies and occupy less area. Techniques to increase the maximum operating frequency of flip-flop based designs, such as time-borrowing, rely on tight hold constraints that are difficult to satisfy using traditional back-end optimization techniques. We propose *Mix & Latch*, a methodology to increase the operating frequency of synchronous digital circuits using a single clock tree and a mixed distribution of positive- and negative-edge-triggered flops, and positive- and negative-level-sensitive latches. An efficient mathematical model is proposed to optimize the type and location of the sequential elements of the circuit. We ensure that the initial registers are not moved from their initial location, although they may change type, thus allowing the use of equivalence checking and static timing analysis to verify formally the correctness of the transformation. The technique is validated using a 28 nm CMOS FDSOI technology, obtaining 1.33 X post-layout average operating frequency improvement on a broad set of benchmarks over a standard commercial design flow. Additionally, the circuit area was also reduced by more than 1.19 X on average for the same benchmarks, although the overall area reduction is not a goal of the optimization algorithm. To the best of our knowledge, this is the first work that proposes combining mixed-polarity flip-flops and latches to improve the circuit performance.

INDEX TERMS Integrated circuit synthesis, design automation, sequential circuits, latches, flip-flops.

I. INTRODUCTION

Sequential circuits use flip-flops (FFs) or latches for data storage. Latches can be used in error-resilient applications [1], work at lower supply voltages, reduce power consumption [2], [3], [4], and can increase operating frequency [5], [6]. Yet, more complex timing constraints limit their support in commercial flows and their use in industrial designs. This motivated us (and many others, as described below) to automatically convert an FF-based design to a latch-based design. We focus mostly on *performance, i.e., on reducing the clock period*, but in some cases we also achieve *area improvements*. We use three combined techniques: (1) latch-based design with time borrowing, (2) negative transparent latch (NTL)-based retention barriers, instead of delay padding, to meet short-path hold constraints, and (3) only one clock tree. We start with an arbitrary FF-based design and *temporarily* convert it into a pulse-based single-phase positive transparent latch (PTL) design functionally equivalent to the original, but faster thanks to time borrowing and reduced delays in latches. Latch-based designs exacerbate the presence of hold violations, which must traditionally be fixed either by using narrow clock pulses or adding buffers to delay short paths [7]. Narrow clock pulses are difficult to distribute without vanishing and

The associate editor coordinating the review of this manuscript and approving it for publication was Christian Pilato.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

reduce the margins for time borrowing. On the other hand, buffers increase area and power and may have a negative impact on timing.

We propose the *Mix&Latch* method, which uses a conventional 50 % duty cycle (DC) single-phase clock. Hold time violations are solved by inserting NTLs driven by the same clock tree as the PTLs. First *the resulting clock period is optimized* by combining time borrowing and NTL retiming. Then, as a *secondary objective, area recovery is used to reduce the area overhead* by creating NTL-PTL sequences whenever possible. These primary/secondary pairs are then converted into either positive-edge-triggered flops (PETFs) or negative-edge-triggered flops (NETFs), thus obtaining an optimized *mixed* design with PTL, PETF, NTL, and NETF sequential elements. *Mix&Latch* also preserves a sequential element in each of the original FF locations. This enables a 1-to-1 mapping from FFs to sequential elements and ensures that equivalence checking can be performed using conventional methods comparing combinational clouds.

Fig. 1 shows the application of our optimization algorithm to a simple case. The original arbitrary PETF circuit is shown in Fig. 1a. All pins are annotated with a parenthesized number pair indicating the (*minimum arrival time at pin p* ($AT_p^{\min}$)*, maximum arrival time at pin p* ($AT_p^{\max}$)). For simplicity, in this figure we assume unitary delays for combinational gates, zero delays for the sequential elements, and zero setup/hold FF constraints, while our algorithm uses delays from timing analysis. In a PETF-based circuit, the minimum clock period, $T_{\min}$, is set to the longest maximum arrival time at the endpoint pin ($AT_{\text{end}}^{\max}$), hence $T_{\min} = 6$ in our example. After the PTL conversion shown in Fig. 1b, the circuit can use time borrowing up to half a clock period (for $DC = 50\,\%$) for $AT_{\text{end}}^{\max}$

$$AT_{\text{end}}^{\max} = T_{\min}\,(1 + DC) \Longrightarrow T_{\min} = \frac{6}{1 + 0.5} = 4. \qquad (1)$$

Note that there is an additional critical path with delay 6, PTL B → PTL Z, due to time borrowing at PTL B.

Despite the desirable $T_{\min}$ reduction by 33 % compared to the PETF version, there are hold violations at the inputs of PTLs X, Y, and Z because their minimum arrival time ($AT^{\min}$) is lower than the positive pulse width, $PPW = DC \cdot T_{\min}$. To solve the hold violations, a group of nets is selected using the mathematical model described below, and an NTL is placed in front of the endpoint pin of the selected nets. As the NTLs become transparent after the positive pulse, they guarantee a delay longer than the PPW for all paths.

However, the added NTLs can reduce performance. For example, Fig. 1c shows that while placing the NTL too close to the source PTL solves the hold violation at the input of PTL X, the additional delay causes a setup violation at the input of PTL Z, which now requires a longer period, $T > T_{\min}$. Fig. 1d shows that an NTL can solve the hold violations at the input of PTL Z, but it causes a new setup violation at the input of the NTL that closes at T. Hence, the signal cannot reach PTL Z in time, which also requires a longer $T > T_{\min}$.

Our algorithm optimizes the position of NTLs to reach a solution that, as shown in Fig. 1e, solves all hold violations without performance penalty, under the assumptions discussed in Section III-C.

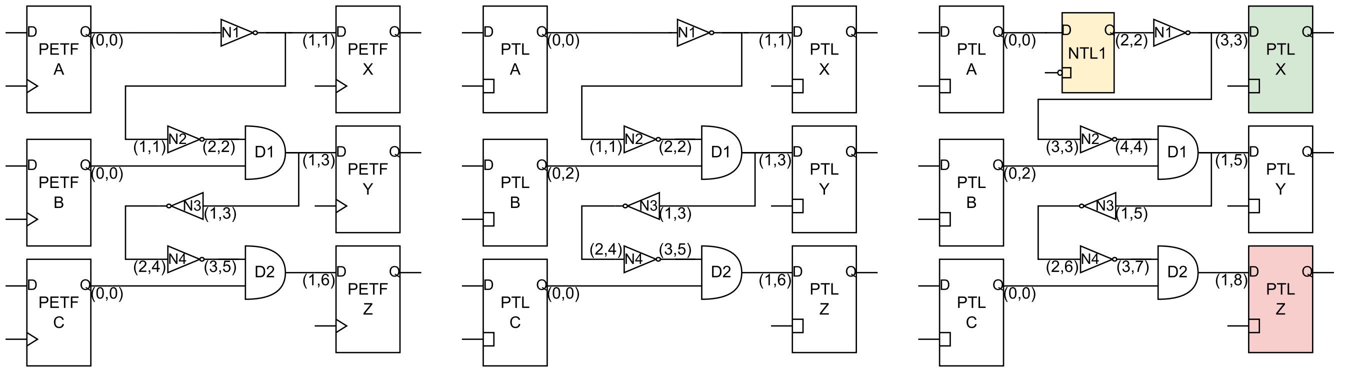
Next, adjacent NTL-PTL pairs are merged as PETF to reduce the area, and PTL-NTL pairs as NETF, as shown in Fig. 1f. This solution has the same $T_{\min}$ and area as the one in Fig. 1b, no hold violations, and uses the same number of sequential elements as the original version. In some cases some latches cannot be merged, which leads to some area penalty (discussed in the experimental results). In other cases the PTLs do not need hold time fixing, yielding both faster and smaller circuits.

Several works propose design optimization using a mix of PETF and PTL. Here we describe the main ones, in order to set the stage for our work, while a more complete literature review is provided in the next section. Hassan et al. [3] propose to start from an FF-based netlist, analyze sequences of three FFs, and replace the middle one with a PTL retimed to match the timing constraints. This approach seems to increase the clock frequency, reduce the power consumption and the cell area, but the experimental data cover only logic synthesis, without considering placement and routing. Furthermore, equivalence checking may be more difficult because retiming changes the original position of the sequential elements [8].

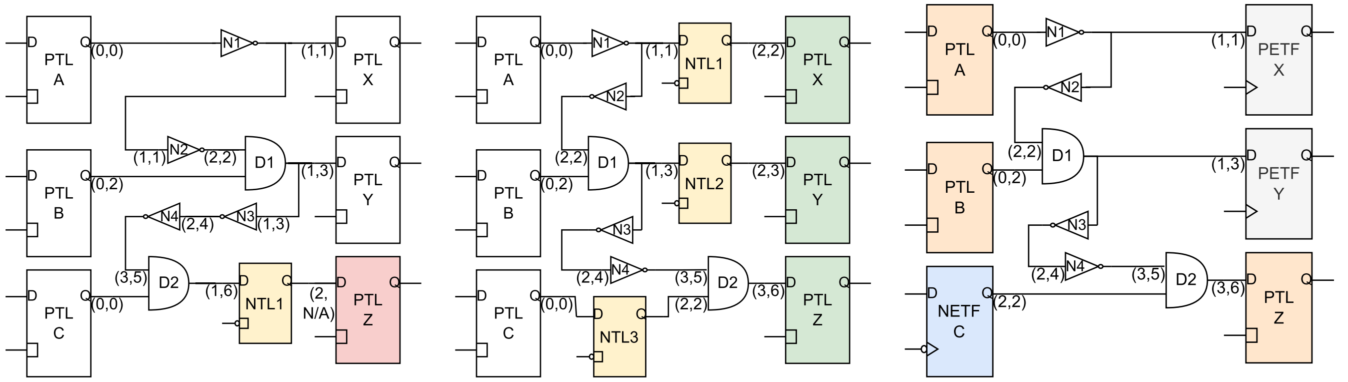
Singh et al. [2] describe a retiming method to generate a PTL-NTL-based netlist starting from a FF-based one. Because the synthesis tools have poor support for latch retiming, they propose replacing the primary/secondary latches with FF pairs, doubling the frequency and finally retiming the design using a commercial tool. Although they focus on reducing the power consumption, the results are poor in terms of both power and area because the algorithm is effective for only one frequency due to a sub-optimal retiming strategy. Moreover, experimental results are shown only for one circuit.

Our main contributions to the state-of-the-art are:

- A two-step implementation flow to obtain a working layout for an optimized version (Fig. 1f) of the PETF-based netlist (Fig. 1a). The implementation is fully based on commercial EDA tools and we fully exploit useful skew, both in the baseline against which we compare and in our own results.
- A methodology to reduce the sequential resources and generate the NTL allocation, using post-layout timing data and exploiting incremental placement and routing starting from the post-layout netlist. The NTLs work as retention barriers for signals in short paths, reducing the hold constraints complexity. To recover area, the PTL-NTL pairs are merged into FFs.
- Maintaining a 1-to-1 correspondence between each original FF and a FF or a latch in the final circuit, to allow equivalence checking for design verification with traditional tools.



(a) Original arbitrary circuit using PETFs (b) PTL-based version of the original circuit (c) NTL close to source PTL



(d) NTL close to sink PTL (e) NTL-based solution (f) Solution with merged resources: PTL-to-NTL yields NETF, NTL-to-PTL yields PETF

FIGURE 1. **Optimization algorithm applied to an arbitrary positive-edge-triggered flop (PETF) circuit (1a). Conversion to a circuit based on positive transparent latch (PTL) (1b). Optimization for sequences of PTL-to-negative transparent latch (NTL) (1c) or NTL-to-PTL (1d). Conversion of complementary latch sequences (1e) into positive-edge-triggered flops (PETFs) (1f) or NETFs. Pins are annotated with (*minimum arrival time (AT^{min}), maximum arrival time (AT^{max})*). Gate delays are unitary and sequential delays are zero.**

- The evaluation of the proposed method on a diverse range of circuits, each implemented at multiple frequencies, achieving an average frequency increase of 1.33 X and a 1.19 X average area reduction.

II. RELATED WORK

A. LATCH-BASED DESIGN TIMING ANALYSIS

The seminal work of [9] and [10] provides a formal definition of the min clock cycle problem for PTL-based circuits based on linear programming. They resolve the non-linearity of the constraints using linear inequality constraints to implement the min/max functions. Their formulation points out two main parameters for optimizing the clock scheduling for each PTL: the phase and the width of the high clock pulse.

B. PULSED LATCH SOLUTIONS

The *pulsed latch* design style is introduced in [11], [12], [13], [14], and [15] using a circuitry to generate, from the input clock of the digital system, a different width of the high clock pulse (T_i) for each sequential block. To limit the area cost, the pulse generators are shared by PTL groups and they are integrated either in a single sequential cell for pulsed FF (P-FF) (pulsed latches with the pulse generator within the latch cell) or in a cell containing multiple sequential blocks for pulsed registers.

Nevertheless, with shared pulse generators it is very difficult to prevent pulse signal degradation in all operating conditions [11], and the additional retimed registers for solving the remaining hold violations further increase the area.

C. SINGLE AND MULTI-PHASE CLOCK SOLUTIONS

A different set of solutions use single or multi-phase clocking schemes without focusing on the width of the high clock pulse. Zhang and Calhoun [16] study the distribution of errors caused by sub-threshold voltage supply and propose a two-phase clocked latch-based method to solve the timing violations. Fojtik et al. [1] analyze a two clock-phase latch-based implementation of Razor flops to detect errors in an ARM Cortex-M3 processor. Cheng et al. [17] discuss a conversion algorithm using three clock phases to improve area and power consumption. Yoshikawa et al. [18] present a single-phase forward retiming algorithm for FF-based design conversion, using commercial tools for retiming. Hassan et al. [3] and Singh et al. [2] present

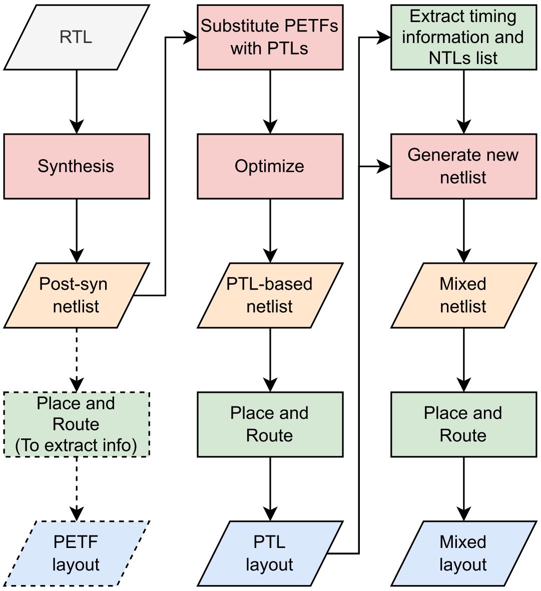


FIGURE 2. Implementation flow starting from the register-transfer level (RTL) description using positive-edge-triggered flops (PETFs), positive transparent latches (PTLs), and negative transparent latches (NTLs). Synthesis steps are in red, post-synthesis netlists in orange, layout steps in green, and post-layout netlists in blue. The PETF layout is only used to provide the baseline results.

implementation flows to transform FF-based designs into latch-based or mixed designs. In almost all previous cases, the optimization uses post-synthesis timing information that may substantially differ from the post-layout one, thus potentially leading to grossly sub-optimal post-layout performance. Furthermore, [3], [18] evaluate the performance only on post-synthesis data, thus ignoring the place and route (P&R) overheads.

III. *Mix & Latch* OPTIMIZATION FLOW

Fig. 2 shows our optimization flow, which starts from a register-transfer level description description and produces a layout with mixed sequential resources. It includes four main steps:

- Generate the PTL-based layout by replacing all sequential elements with PTLs (shown in the second column in Fig. 2 and discussed in Section III-A).
- Create a graph representation of the timing and positional information extracted from the PTL-based layout (discussed in Section III-B, Section III-C, and Section III-D).
- Define the circuit location of NTLs, PETFs, and NETFs using an integer linear programming (ILP) formulation, and inserting them in the PTL-based netlist.
- Generate the layout of this new circuit (see the right column in Fig. 2 and the discussion in Section III-E).

The NTL selection using ILP is similar to backward retiming [18] in a primary/secondary FF netlist. However, the formulation and graph representation are different because they consider the post-layout timing data and avoid the redundant NTLs. The designs are synthesized and implemented at several clock frequencies to determine iteratively the highest possible operating frequency for both the mixed design and of the PETF design.

We leave to future work the in-depth analysis of design for testability (DFT) needed for the practical adoption of our methodology. We note however that DFT can be implemented with traditional tools by adding some scan-only NTLs to the PTLs [19].

Retiming techniques have the drawback that equivalence checking for design verification cannot be solved in a reasonable amount of time even for relatively small circuits, such as the `s38584` from the ISCAS benchmark [8], which we also use in our experiments as shown in Section IV. *Mix & Latch* does not have this problem because it preserves a 1-to-1 correspondence with the FFs in the original design using either FFs or PTLs. The 1-to-1 correspondence also helps solving the initialization problem for the netlist, i.e., finding a consistent initial value of the circuit registers that maintains the circuit equivalence [20].

A. POSITIVE TRANSPARENT LATCH-BASED CIRCUIT

The first processing step generates the PTL-based layout. The register-transfer level (RTL) description of the target design is synthesized using a commercial tool. The considered circuits have only PETFs to ease the analysis, but the same methodology can be extended to circuits based on NETFs or mixed. Once the netlist is synthesized, all the PETFs are replaced with PTLs using the same commercial tool. Because cell resizing will be automatically done by the layout tool (if needed), the PETFs are replaced with the smallest PTLs from the technological libraries.

The netlist modified this way is provided to the layout tool, which produces the post-P&R design. Unlike [9], [10], all hold constraints are temporarily ignored (using a standard design constraint command of the tool) to obtain a layout of the PTL-based netlist that meets the setup constraints.

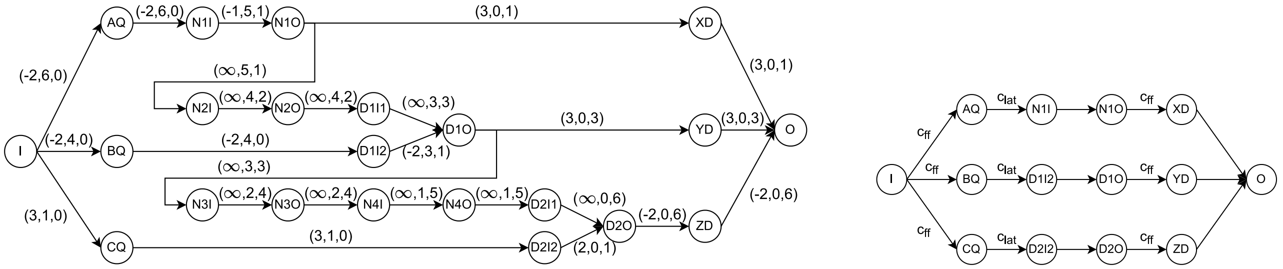
The generated layout thus potentially violates hold conditions, which will be solved afterwards.

B. GRAPH MODEL

The state-of-the-art circuit graph representations [10], [21] are not suitable for our optimization algorithm because they either exclude the sequential elements [21], or aggregate pin data for the worst case delay [10].

For our method, the circuit is represented as a graph (V, E), where V represents the set of all pins and I/O ports and E the connections (wires or cells) between them. The nets and pins of the clock tree are not included.

Fig. 3 shows an example of two graphs which are discussed later. Static timing analysis (STA) timing information is a three-value tuple associated to graph edges (see Section III-C), while latch location is a value associated to edges of a different graph (see Section III-D).


AQ
(-2,6,0)
N1I
(-1,5,1)
N1O
(3,0,1)
XD
(-2,6,0)
(∞,5,1)
(3,0,1)
N2I
(∞,4,2)
N2O
(∞,4,2)
D1I1
(∞,3,3)
D1O
(3,0,3)
YD
(3,0,3)
O
I
(-2,4,0)
BQ
(-2,4,0)
D1I2
(-2,3,1)
(∞,3,3)
N3I
(∞,2,4)
N3O
(∞,2,4)
N4I
(∞,1,5)
N4O
(∞,1,5)
D2I1
(∞,0,6)
(-2,0,6)
(-2,0,6)
ZD
(3,1,0)
D2O
CQ
(3,1,0)
D2I2
(2,0,1)
I
c_ff
AQ
c_lat
N1I
N1O
c_ff
XD
c_ff
BQ
c_lat
D1I2
D1O
c_ff
YD
O
c_ff
CQ
c_lat
D2I2
D2O
c_ff
ZD

(a) : Timing Graph (TG)

(b) : Short-Path Graph (SPG)

FIGURE 3. Graph generation: (a) TG of the example in Fig. 1b. The attribute is a 3-tuple with elements computed using Alg. 1, Eqn. (6), and Eqn. (5), respectively. The first value is an estimation of the setup slack caused by NTL insertion, the second and third values are the lengths of the sub-paths generated by NTL insertion. (b) SPG of the same circuit: it has fewer edges and vertices than TG because it considers only pins and connections that belong to short paths.

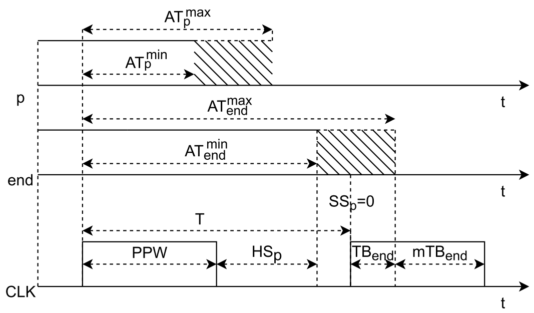

AT_p^max
AT_p^min
p
t
AT_end^max
AT_end^min
end
t
SS_p=0
T
PPW
HS_p
TB_end
mTB_end
CLK
t

FIGURE 4. Timing diagram, showing the arrival times and the slacks related to an ideal clock. During attribute computations the clock latencies are real and referred to the clock tree built in the PTL-based netlist. Because $AT_{\text{end}}^{\max}$ arrives during the PTL transparent window, SS_p is 0.

C. TIMING GRAPH

We create a timing graph (TG) that drives our optimization algorithm to limit the setup slack (SS) degradation due to potential NTL insertions before gate input pins in the PTL-based netlist. The computation of the edge attributes of this graph uses the timing data extracted from static timing analysis (STA), shown in Fig. 4, Fig. 5, and Table 1. Unlike Fig. 4, all arrival times are obtained from the STA considering the clock latencies of the PTL-based layout. The pin (p) for which STA extracts the info is the endpoint of the edge (e) to which the related attribute is associated. From the STA timing data we obtain three values: (1) the *Estimated Setup Slack for pin p* (ESS_p), (2) the *p to SCP$_p$ endpoint delay* (D_{ptl}^{p}), and (3) the *SCP$_p$ startpoint to p delay* (D_p^{ptl}). These three values are assigned as a 3-tuple attribute to the edges of TG.

1) ESTIMATED SETUP SLACK

The Estimated Setup Slack (ESS_p) is computed for each edge (e) endpoint pin (p) using Algorithm 1, which estimates the value of the setup slack (SS) related to the pin (p) if an NTL were placed in front of it. It receives in input the timing info from STA for the considered pin and returns the attribute ESS_p. It is important to highlight that the computation of

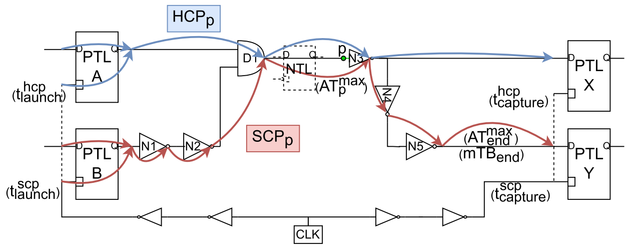

HCP_p
PTL A
D1
p
NTL
N3
(AT_p^max)
N4
(t_launch^hcp)
PTL B
N1
N2
SCP_p
N5
(AT_end^max)
(mTB_end)
PTL X
(t_capture^hcp)
PTL Y
(t_launch^scp)
(t_capture^scp)
CLK

FIGURE 5. Circuit showing how the information from the STA tool is extracted.

TABLE 1. Timing computed by Algorithm 1 with data from static timing analysis.

NAME	DEFINITION
HCP$_p$	Hold critical path passing through pin p
SCP$_p$	Setup critical path passing through pin p
$t_{\text{launch}}^{\text{HCP}}$	Clock latency of the considered HCP$_p$ launching PTL
$t_{\text{launch}}^{\text{SCP}}$	Clock latency of the considered SCP$_p$ launching PTL
$AT_p^{\max}$	Maximum arrival time at pin p
$AT_p^{\min}$	Minimum arrival time at pin p
SS_p	Setup slack at pin p
HS_p	Hold slack at pin p
$t_{\text{capture}}^{\text{HCP}}$	Clock latency of the considered HCP$_p$ endpoint PTL
$AT_{\text{end}}^{\max}$	Maximum arrival time at endpoint PTL
$AT_p^{\min}$	Minimum arrival time at endpoint PTL
mTB_{end}	Time borrowing margin at endpoint PTL of SCP$_p$
TB_{end}	Time borrowing at endpoint PTL of SCP$_p$
PPW	Clock positive pulse width

$AT_p^{\max}$ takes into account the possible time borrowed by the launching PTL. It relies on the following assumptions:

- To estimate the SS_p degradation caused by the NTL insertion, we need the NTL opening time, $t_{\text{open}}^{\text{ntl}}$, and closing time, $t_{\text{close}}^{\text{ntl}}$. They depend on the NTL clock latency ($t_{\text{del}}^{\text{ntl}}$), from PPW and from T. Since it is difficult to know $t_{\text{del}}^{\text{ntl}}$ at this stage, we assume that it is equal to $t_{\text{capture}}^{\text{HCP}}$, unless an NTL is merged into a NETF when we use the latency $t_{\text{launch}}^{\text{HCP}}$.

Algorithm 1 Estimated Setup Slack Attribute for Pin p

Inputs: Parameters from Table 1 **Output:** ESS_p

```
1:  if $HS_p \geq 0$ then
2:      $ESS_p \leftarrow \infty$
3:  else
4:      if $p$ is output of PTL then
5:          $t_{\text{del}}^{\text{ntl}} \leftarrow t_{\text{launch}}^{\text{HCP}}$
6:      else
7:          $t_{\text{del}}^{\text{ntl}} \leftarrow t_{\text{capture}}^{\text{HCP}}$
8:      end if
9:      $t_{\text{open}}^{\text{ntl}} \leftarrow t_{\text{del}}^{\text{ntl}} + PPW$
10:     $t_{\text{close}}^{\text{ntl}} \leftarrow T + t_{\text{del}}^{\text{ntl}}$
11:     if $AT_p^{\max} < t_{\text{open}}^{\text{ntl}}$ then
12:         $ESS_p \leftarrow SS_p - t_{\text{open}}^{\text{ntl}} + AT_p^{\max} + mTB_{\text{end}}$
13:     else
14:         if $AT_p^{\max} > t_{\text{close}}^{\text{ntl}}$ then
15:             $ESS_p \leftarrow t_{\text{close}}^{\text{ntl}} - AT_p^{\max}$
16:         else
17:             $ESS_p \leftarrow SS_p + mTB_{\text{end}}$
18:         end if
19:     end if
20: end if
```

In Fig. 4, the NTL would have the clock latency of PTL X. Lines 4–10 implement these computations. The condition on line 4 checks if the pin is the output of a PTL, thus the resulting NTL would be merged into a NETF.

- The additional delay from NTL insertion is ignored because it is usually small compared to the SCP$_p$ delay and because it is hard to estimate before the layout. Note that we ignore it only to simplify the Timing Graph (TG) generation, but in the final layout step the P&R tool does consider the NTL delays.

We explain the steps in Algorithm 1 analyzing the four cases which cover all the possible combinations, while in Fig. 3a we illustrate an example of TG for the circuit of Fig. 1b:

a: CASE 1 – POSITIVE HOLD SLACK

If the considered p has positive hold slack, HS_p, then there is no violation to fix. To reduce the number of NTLs that will be used after retiming, all the NTLs that would be placed close to pins not belonging to short paths will not be added to the PTL netlist. Avoiding NTL insertion means no SS_p degradation, hence in this case we set weight (W) to ∞ (lines 6–7 of Algorithm 1). An example is the edge $D1O \rightarrow N3I$ in Fig. 3a, corresponding to the edge $D1 \rightarrow N3$ in Fig. 1b, which does not belong to a short path.

b: CASE 2 – NTL CLOSE TO THE SOURCE PTL

The additional delay caused by the late opening of the NTL may cause a setup violation, as shown in Fig. 1c. Attribute computation estimates the degradation of the pin setup slack, taking into account the late arrival time at the selected pin ($AT_p^{\max}$), SS_p, $t_{\text{open}}^{\text{ntl}}$, and the margin for time borrowing (mTB_{end}). The delay introduced by the NTL can be tolerated up to mTB_{end}. Lines 11–12 of Algorithm 1 perform these computations. An example is edge $AQ \rightarrow N1I$ from Fig. 3a, corresponding to the edge from PTL A to $N1$ in Fig. 1c.

Considering that the SCP for this edge ends in PTL Z, the parameters $SS_p, AT_p^{\max}$ and mTB_{end} are all equal to 0 because the SCP delay is equal to the T added to the maximum time borrowing. $t_{\text{open}}^{\text{ntl}}$ is equal to 2 for all the cases shown in Fig. 3a because the clock is considered ideal. Given the previous considerations, compute ESS_p:

$$ESS_p = 0 - 2 + 0 + 0 = -2 \tag{2}$$

c: CASE 3 – NTL CLOSE TO THE SINK PTL

If the input signal of the sink PTL belongs to a critical path, then the setup constraints added by the early NTL closing will likely prevent satisfying the setup constraints. If the late arrival time at the pin, $AT_p^{\max}$, exceeds $t_{\text{close}}^{\text{ntl}}$, then the signal will not pass through the NTL. The SS_p degradation is computed as the difference between these two values (lines 14–15 of Algorithm 1). An example is edge $D2O \rightarrow ZD$ from Fig. 3a, corresponding to the edge from $D2$ to PTL Z in Fig. 1d. $t_{\text{close}}^{\text{ntl}}$ is equal to 4 for all the cases because the clock is considered ideal and $AT_p^{\max}$ is 6. Given the previous considerations, ESS_p is computed as:

$$ESS_p = 4 - 6 = -2 \tag{3}$$

d: CASE 4 – GENERAL CASE

If none of the previous cases occurs, then $AT_p^{\max}$ at the NTL input falls into the NTL transparency interval and there is no SS_p degradation (line 17 of Algorithm 1). An example is edge $D1O \rightarrow YD$ from Fig. 3a, corresponding to the edge from $D1$ to PTL Y in Fig. 1d.

Considering that the SCP for this edge ends in PTL Y, $SS_p = 1$ because the signal arrives 1 time unit before the rising edge of the clock, while $mTB_{\text{end}} = 2$ because there is no time borrowing. Given the previous considerations, ESS_p can be computed as:

$$ESS_p = 1 + 2 = 3 \tag{4}$$

2) SUB-PATH DELAYS

The second value of the tuple, D_{ptl}^p, shows the delay of the path between the pin p and the endpoint PTL of SCP$_p$. It is equal to the difference between $AT_{\text{end}}^{\max}$ and $AT_p^{\max}$

$$D_{\text{ptl}}^p = AT_{\text{end}}^{\max} - AT_p^{\max}. \tag{5}$$

The third value of the tuple, D_p^{ptl}, shows the delay of the path between the start point PTL of the SCP$_p$ and the pin p. It is computed as the difference between $t_{\text{launch}}^{\text{SCP}}$ and $AT_p^{\max}$

$$D_p^{\text{ptl}} = AT_p^{\max} - t_{\text{launch}}^{\text{SCP}}. \tag{6}$$

TABLE 2. Variable definitions for Alg. 2. (ILP model).

NAME	DEFINITION
SPG	Short-Path Graph
T	Cycle period
δ	Fraction of T to meet setup constraints
c_{ff}	Cost of merging an NTL with a PTL
c_{lat}	Cost of not merging an NTL ($c_{\text{lat}} > c_{\text{ff}}$)
E_{cells}	Edges between pins of the same cell
E_{wires}	Edges between pins of different cells
E_{ff}	Edges where NTL would be merged in PETF/NETF
E_{lat}	Edges where NTL would not be merged
$R(e)$	Edge selection value

D. SHORT-PATH GRAPH

The Short-Path Graph (Short-Path Graph (SPG)) is a subgraph of the TG that only contains the pins and edges that belong to short paths, i.e., all those pins p such that $HS_p < 0$. Hold violations will be fixed by finding a cut (subset of edges) of the SPG where the NTLs will be inserted.

Two types of edges can be distinguished in the SPG:

$$E = E_{\text{cells}} \cup E_{\text{wires}}$$

whereas E_{cells} correspond to those edges that connect input-to-output pins in combinational cells and E_{wires} correspond to the remaining edges. The cut of the SPG must be defined using edges in E_{wires}.

The insertion of an NTL in an edge may benefit from the presence of an adjacent PTL at the start or end point of the edge. Thus, both latches can be merged into an FF, either PETF (NTL-PTL) or NETF (PTL-NTL), as shown in Fig. 1f. Thus, we can define

$$E_{\text{wires}} = E_{\text{ff}} \cup E_{\text{lat}}$$

to distinguish these edges, with E_{ff} representing the edges in which the merging is possible and E_{lat} representing the remaining edges. Additionally, two parameters are defined to represent the cost of inserting an NTL, c_{ff} and c_{lat}, with $c_{\text{ff}} < c_{\text{lat}}$, since merging implies area savings. These parameters can be tuned to control the area overhead of the solution.

Graph 3b shows the SPG of the example circuit from Fig. 1b.

E. INTEGER LINEAR PROGRAMMING MODEL

Starting from the SPG and the attributes computed from static timing analysis of the PTL post-layout netlist, an ILP model is defined to fix the hold violations and select the NTL locations. Alg. 2 and Tab. 2 show the ILP model and the definition of the algorithm variables.

For each pin (p) of the SPG, a binary variable p is created. For each edge (e), $p_{\text{end}}(e)$ and $p_{\text{start}}(e)$ represent the variables associated to the endpoint and the start point of e, respectively. Each edge is characterized by the edge selection value, $R(e)$, defined as

$$R(e) = p_{\text{end}}(e) - p_{\text{start}}(e). \tag{13}$$

Algorithm 2 Integer linear programming (ILP) Model

```
Inputs:  SPG, $T$, $\delta$, $c_{\text{ff}}$, $c_{\text{lat}}$
Output:  Location of the NTLs (edges with $R(e) = 1$)
$E \leftarrow$ Edges($SPG$)
$E_{\text{cells}}, E_{\text{wires}}, E_{\text{ff}}, E_{\text{lat}} \leftarrow E$
$ESS_p, D_p^{\text{ptl}}, D_{\text{ptl}}^p \leftarrow$ TimingAttributes($E$)
```

$$\text{minimize} \quad c_{\text{lat}} \sum_{\forall e \in E_{\text{lat}}} R(e) \; + \; c_{\text{ff}} \sum_{\forall e \in E_{\text{ff}}} R(e) \tag{7}$$

$$\text{subject to} \quad \forall e \in E_{\text{cells}} : \quad R(e) = 0 \tag{8}$$

$$\forall e \in E_{\text{wires}} : \quad R(e) \geq 0 \tag{9}$$

$$\forall e \in E_{\text{wires}} : \quad R(e) \cdot ESS_p(e) \geq 0 \tag{10}$$

$$\forall e \in E_{\text{wires}} : \quad R(e) \cdot D_{\text{ptl}}^p(e) \leq \delta \cdot T \tag{11}$$

$$\forall e \in E_{\text{wires}} : \quad R(e) \cdot D_p^{\text{ptl}}(e) \leq \delta \cdot T \tag{12}$$

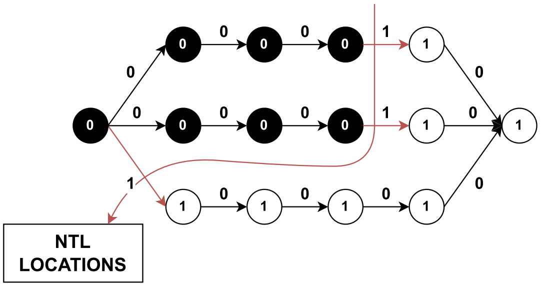


FIGURE 6. Solution example showing the cut chosen for the SPG from Fig. 3b. The vertex attributes correspond to the *P* variables of the ILP model and the edge attributes represent the edge selection value (*R*(*e*)) computed for each edge. The vertices with input edge attribute equal to 1 are selected for NTL insertion.

The cut (location of the NTLs) is defined for those edges with $R(e) = 1$, i.e., $p_{\text{start}}(e) = 0$ and $p_{\text{end}}(e) = 1$, as shown in Fig. 6.

The cost function (7) accounts for the number of new sequential elements added to the circuit, i.e., the number of NTLs inserted in edges not connected to a PTL. This will push the solution of Algorithm 2 to use as many NETFs and PETFs as possible to reduce the final number of sequential elements in the circuit.

The constraint (8) avoids that Alg.2 selects edges representing connections between pins of the same cells (E_{cells}).

The constraint (9) enforces $p_{\text{end}}(e) \geq p_{\text{start}}(e)$, because $p_{\text{end}}(e)$ and $p_{\text{start}}(e)$ are binary this restricts $R(e)$ to be binary. It also implies that all pins p belonging to a path that reaches $p_{\text{start}}(e)$ will have $p = 0$, while all pins belonging to a path that crosses $p_{\text{end}}(e)$, reaches p, and ends at a PTL will have $p = 1$. Then, the algorithm splits the graph in two partitions, before and after the NTLs, by removing the edges with $R(e) = 1$. The partition in which all pins have $p = 1$, i.e. the part of the graph that includes the PTL endpoints, will have no early arriving signals. Fig. 6 shows an example of the graph partitioning generated by the model.

Although solving an ILP generally has very high runtime, in this particular case it is very close to a max-flow min-cut problem, which is known to have polynomial complexity. This is the likely reason why the runtime of our algorithm remains very small, as shown in Table 4, even for designs with tens of thousands of gates and FFs. The development of a heuristic algorithm is left to future work, if the execution time becomes excessive, e.g. comparable to or larger than the physical design time.

The constraint (10) guides the model towards solutions that do not worsen setup violations, because the SS_p for each selected edge for NTL insertion must be greater than zero. The estimation done in ESS_p is an approximation of the final SS_p that takes into account not only the length of the combinational logic delay, but also the clock tree latency generated by the layout tool, as discussed in Section III-C.

However, this is an approximation and we need two more inequalities, (11,12), to simplify the problem of meeting the setup constraints. The D^p_{ptl} and D^{ptl}_p attributes report the distance, in terms of post-layout delay, between each pin p and the source/sink PTLs. An NTL placed in front of p divides the path in two parts and the two graphs give an estimation of the length of these sub-paths. To make these paths as short as possible, these time intervals are constrained to be a fraction δ of T, that is a parameter of our algorithm. The value of δ, with $0 < \delta < 1$, is discussed in the next section.

IV. EXPERIMENTAL RESULTS

Open-source `PULP` [22] library is used to model the ILP, the default solver is `CBC`. To evaluate the proposed algorithm, we apply the optimization flow to 13 circuits from a pool of benchmarks, each implemented at a range of operating frequencies. Four circuits are cryptographic IPs from the CEP benchmark [23], eight are from the ISCAS89 benchmark [24], and one is a small processor core from the ITC99 benchmark [25]. The implementation flow uses an industrial 28 nm FDSOI CMOS technology, Design Compiler from Synopsys for logic synthesis, and Innovus from Cadence for P&R.

We set $\delta = 0.75$ in Algorithm 2, i.e. the maximum sub-path delay is 75 % of T. Since δ defines the length of the sub-paths generated by NTL insertion, 75 % for a DC of 50 % means that the two sub-paths are reasonably balanced. Further exploration of the impact of δ is left to future work.

We also set $c_{\text{ff}} = 0$ and $c_{\text{lat}} = 1$ to account for the number of new sequential elements in the circuit.

Table 3 shows the frequency improvement for the considered benchmarks, together with the final sequential resource mix. The average improvement in frequency is about 1.33X. We used a granularity of 0.1 ns in the exploration of the minimum clock period (T). The algorithm is doing better than average for the cryptography IPs like `des3` and `md5`, probably because they are designs with acyclic paths that are generally not well-balanced.

TABLE 3. Operating frequency and sequential resources for designs from ISCAS$^\diamond$, CEP$^\circ$ and ITC99$^\bullet$ benchmarks. Columns labeled 'Original" refer to PETF-based layouts, while those labeled "mixed" refer to the optimized ones.

Design	f_{max} (GHz)			Original	Mixed			
	Original	Mixed	Ratio	*PETF*	*PETF*	*NETF*	*PTL*	*NTL*
s1196$^\diamond$	2.50	3.33	1.33	18	1	0	17	19
s1423$^\diamond$	2.00	2.50	1.25	74	5	4	65	128
s5378$^\diamond$	1.67	2.00	1.20	176	80	3	93	105
s9234$^\diamond$	2.00	2.50	1.25	145	47	28	70	113
s13207$^\diamond$	1.00	1.43	1.43	625	395	137	93	521
s15850$^\diamond$	1.25	1.67	1.33	442	94	88	260	453
s38417$^\diamond$	0.48	0.67	1.40	1564	690	110	764	1213
s38584$^\diamond$	0.43	0.53	1.21	1275	636	136	503	1116
b22$^\bullet$	0.48	0.67	1.40	613	78	72	463	1141
des3°	0.67	1.00	1.50	199	34	65	100	125
md5°	0.43	0.67	1.53	269	71	61	137	519
sha256°	0.56	0.62	1.12	1040	502	284	254	579
aes_192°	$*^1$	0.33	$*^1$	9382	0	9153	229	530

TABLE 4. ILP execution time (s) and layout times (s). Orig layout refers to the starting PETF netlist, PTL layout to the netlist without hold constraints and with only PTLs, and mixed layout to the final step after NTL insertion. The columns #SEQ. and #COMB. report the number of sequential and combinational elements in the PTL layout, which is the netlist analyzed and provided to the ILP solver.

Design	Orig layout (s)	PTL layout (s)	#SEQ.	#COMB.	ILP (s)	Mixed layout (s)
s1196	461	425	18	332	1	461
s1423	698	577	74	456	1	670
s5378	1078	925	176	645	5	1219
s9234	620	554	145	503	2	566
s13207	4867	4151	625	1918	11	5049
s15850	816	784	442	1589	6	924
s38417	1252	1651	1564	4049	24	5155
s38584	1624	1809	1275	8058	32	1506
b22	2044	2306	613	11 026	29	2242
des3	727	746	199	1795	7	725
md5	4175	3838	269	10 639	19	2114
sha256	3456	2406	1040	4116	30	3121
aes_192	$*^1$	39 703	9382	130 264	3130	44 800

Fig. 7 shows the frequency improvement and the area comparison considering the maximum frequency for the original design and the optimized one. In most of the cases there is an area penalty which can exceed 1.2 X. However, this is compensated by a maximum frequency increase above 1.2 X for these designs. There are also cases in which the performance improvements do not cause any area increase, like for `des3`, `sha256`, `s38584` and `b22`.

Fig. 8 shows the results obtained at frequencies at which both design versions meet the timing for a meaningful area comparison. To demonstrate the actual scalability of this approach, Table 4 shows the runtime of the ILP algorithm compared to the time needed for the layout in the three cases. The ILP runtimes are always less than 10% of the layout times.

A. TIMING CLOSURE

The P&R tool converges to a good solution if, at the end of the automated implementation flow, the hold and setup violations are small and can be fixed with only a few iterations of the final design optimization commands. If they are too

---

[1] The maximum frequency reached for the original designs is low compared to [17] and to the mixed result. For this reason, we do not report it for the frequency and runtime comparisons.

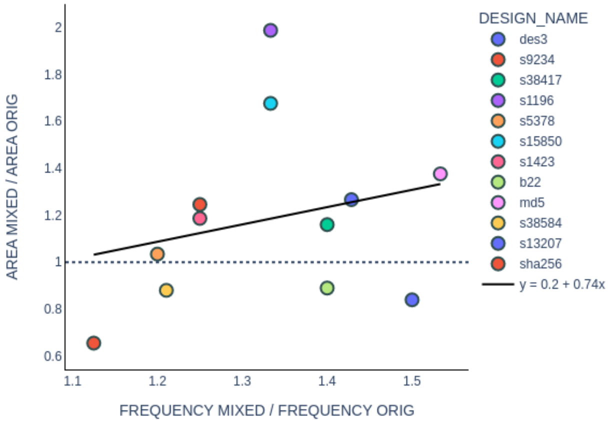


FIGURE 7. **Ratio of post-layout area, considering the layouts obtained at the highest working frequencies for both MIXED and ORIG versions, compared to the related frequency improvements. The black line shows the linear regression of the area increase with respect to the frequency gain. The offset and the slope of the line are stated in the legend.**

large, then the designers typically conclude that the P&R tool cannot implement the design at that specific frequency. In these cases we do not report the area because it is usually excessive. We do at most five optimization iterations to solve the remaining setup and hold violations.

B. AREA COMPARISON

Fig. 8a shows the area comparison at different frequencies in the cases in which both the FF-based and the mixed designs meet the timing. The optimized designs from the ITC99 and CEP benchmarks also have a smaller area than the original ones. However, this is not true in general for the circuits from the ISCAS89 benchmark.

Fig. 8b shows the same area comparison as Fig. 8b, but in this case the x-axis shows the ratio of FFs in the mixed design compared to the original netlist

$$\frac{FF\_MIXED}{FF\_ORIG} = \frac{\max(\#PETF_{\mathrm{MIXED}}, \#NETF_{\mathrm{MIXED}})}{\#PETF_{\mathrm{ORIG}}} \tag{14}$$

where $\#PETF_{\mathrm{ORIG}}$ is the number of PETF in the original circuit. The paths that constrain the design the most are those between pairs of same polarity FFs, because paths from PETF to NETF allow time borrowing and paths from NETF to PETF cannot be generated by Algorithm 2. This is why in (14) we consider the maximum between the two FF types, rather than the sum.

Fig. 8b shows that the area increase in the mixed designs is well correlated with the ratio FF_MIXED/FF_ORIG.

In some circuits, even our cost function, which drives the solution to use as many FFs as possible, could lead to considerable overhead of the mixed version area. Fig. 8c and Fig. 8d show the sequential and combinational area comparison. The sequential area increases in most examples because of the higher number of sequential elements in the design. However, for the designs with a low FFs ratio, easier timing convergence reduces the number of high speed gates. Thus, it tends to compensate this overhead and sometimes leads to a smaller total area. In the next section, we analyze the effect on the area overhead of modifying the NETF allocation cost in the ILP model. We show that results in a significant improvement in the worst cases. We conjecture that power would also be improved, but its evaluation is outside the scope of this paper, which focuses on *performance gains* with limited area cost, or even with area improvement.

C. ALGORITHM TUNING TO REDUCE AREA OVERHEAD

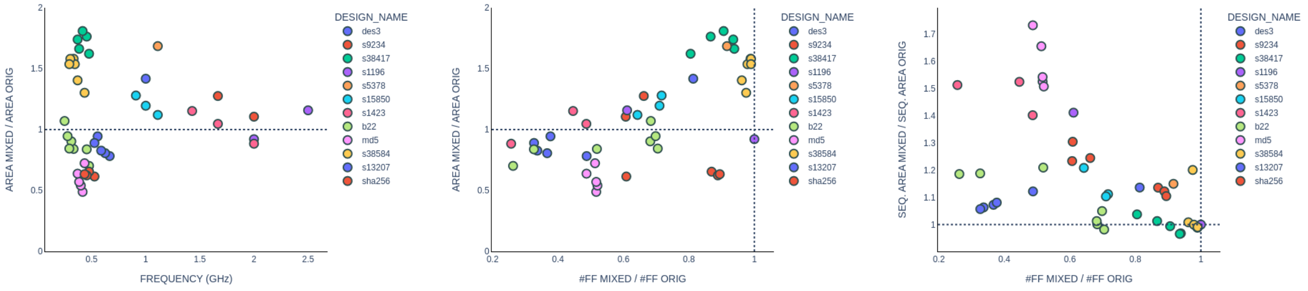
To reduce the area overhead, we discouraged the use of NETFs by increasing the cost of inserting NTLs in locations enabling the PTL-NTL merging.

We slightly modified the ILP model by defining a different cost for merging latches into NETF (cost 1) or PETF (cost 0). Fig. 8e shows that the original area overhead for the ISCAS89 circuits is reduced.

Although this configuration improves the quality of the ISCAS89 worst cases, it increases the area compared to Fig. 8a for some of the `des3`, `md5`, and `b22` designs. Considering the best result among these two values for NETF costs, the average area improvement is 1.19 X over the considered benchmarks, with above-average performance for the cryptography IPs. In some cases belonging to ISCAS89 benchmark, the area increases considerably. Addressing this issue, e.g. by further tuning the algorithm parameters, is left to future work.

D. COMPARISON WITH OTHER WORK

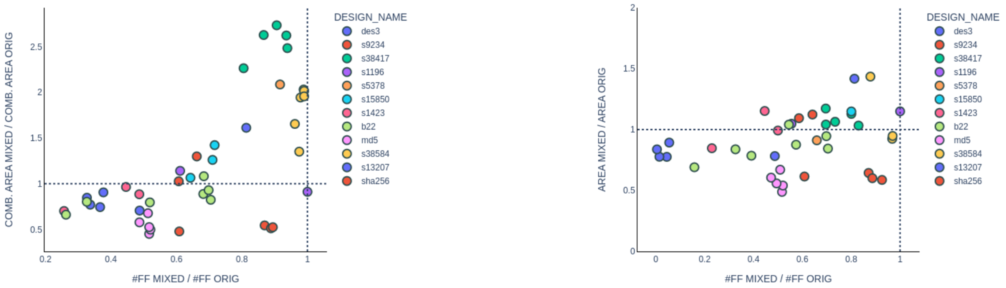
Some of our results can be directly compared with those presented in [17], which converts an FF-based netlist to a 3-phase PTL-based netlist using two variants of the same algorithm. While our main goal is to improve the maximum operating frequency, [17] focuses instead on reducing the area occupation. For this reason, despite the differences in technology and implementation setup, the area overhead introduced by our optimization algorithm is compared with the results of [17]. There are six common benchmark circuits used by us and [17], four from CEP and two from ISCAS89. For the CEP benchmarks, [17] reports maximum area reductions at 500 MHz of 14 % for `des3`, 17.7 % for `sha256`, and 5.8 % for `md5`. Our results in Fig. 8d and Fig. 8e show that, for the cryptography IPs our area reduction exceeds [17], with peaks of 22.38 % for `des3`, 41.32 % for `sha256`, and 51.13 % for `md5`. But for most ISCAS89 benchmark circuits our algorithm increases or only slightly reduces the area, while the area reduction achieved by [17] is more than 10 %. Specifically, our algorithm reduces the area by 5.29 % for `s1423`, 7.59 % for `s5378`, and 8.82 % for `s38584`, and increases the area by 3.28 % for `s38417`, 9.35 % for `s9234`, 4.79 % for `s1196`, and 41.75 % for `s13207`. Note that performance, which is our main design goal, is improved in all cases.



(a) Area vs. operating frequency

(b) Area vs. flip-flops used

(c) Sequential area vs. flip-flops used



(d) Combinational area vs. flip-flops used

(e) Area vs. flip-flops used with modified NETF cost

FIGURE 8. Results of area comparison when both the mixed-based netlist and the PETF-based one successfully yield a layout.

V. CONCLUSION AND FUTURE WORK

The *Mix & Latch* methodology introduced in this paper optimizes flip-flop (FF)-based netlists by replacing the positive-edge-triggered flops (PETFs) with positive transparent latches (PTLs), and solving the hold violations generated by such replacement using an efficient integer linear programming (ILP) model that selects a specific group of edges, and places negative transparent latches (NTLs) on short paths. The algorithm takes as input the timing data from the post-layout netlist of the FF-based design.

We obtain *simultaneously smaller area and higher working frequency* for all the circuits that we considered, except for `s38417`, `s9234`, `s1196`, and `s13207`, where *only performance is significantly improved*. For most cryptography circuits, the area improvement exceeds 1.3 X.

Even though our approach does not aim at improving area and uses just one clock phase, our area reduction is in some cases comparable to that of a recent work [17], which uses three clock phases,rather than just one, and focuses on area optimization. Note also that [17] reports the final area without including the three clock trees, that most likely would decrease their gains significantly.

Future work will focus on further improving parameter selection (e.g., depending on circuit topology) to reduce area occupation at a given operating frequency, on extending the evaluation to other circuits, and on analyzing the impact on power consumption and on design for testability.

REFERENCES

[1] M. Fojtik, D. Fick, Y. Kim, N. 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, Jan. 2013.

[2] K. Singh, H. Jiao, J. Huisken, H. Fatemi, and J. P. de Gyvez, "Low power latch based design with smart retiming," in *Proc. 19th Int. Symp. Quality Electron. Design (ISQED)*, Mar. 2018, pp. 329–334.

[3] N. A. N. Hassan, A. B. A. Manaf, and L. C. Ming, "Optimization of circuitry for power and area efficiency by using combination between latch and register," in *Proc. IEEE Int. Conf. Comput. Appl. Ind. Electron. (ICCAIE)*, Dec. 2011, pp. 240–244.

[4] M. Pons, T.-C. Le, C. Arm, D. Severac, J.-L. Nagel, M. Morgan, and S. Emery, "Sub-threshold latch-based icyflex2 32-bit processor with wide supply range operation," in *Proc. 46th Eur. Solid-State Device Res. Conf. (ESSDERC)*, Sep. 2016, pp. 33–36.

[5] P. A. Hurst and K. R. Brayton, "The advantages of latch-based design under process variation," in *Proc. IWLS*, 2006, pp. 241–246.

[6] B. Taskin and I. S. Kourtev, "Time borrowing and clock skew scheduling effects on multi-phase level-sensitive circuits," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2004, p. 617.

[7] N. V. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Minimum padding to satisfy short path constraints," in *Proc. Int. Conf. Comput. Aided Design (ICCAD)*, 1993, pp. 156–161.

[8] C. Yu, C.-C. Huang, G.-J. Nam, M. Choudhury, V. N. Kravets, A. Sullivan, M. Ciesielski, and G. De Micheli, "End-to-end industrial study of retiming," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2018, pp. 203–208.

[9] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun, "Optimal clocking of synchronous systems," in *Proc. ACM Int. Workshop Timing Issues Specification Synth. Digit. Syst.* Vancouver, BC, Canada: Univ. British Columbia, Aug. 1990.

[10] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun, "Analysis and design of latch-controlled synchronous digital circuits," *IEEE Trans. Comput.-Aided Design Integr.*, vol. 11, no. 3, pp. 322–333, Mar. 1992.
[11] Y. Shin and S. Paik, "Pulsed-latch circuits: A new dimension in ASIC design," *IEEE Des. Test Comput.*, vol. 28, no. 6, pp. 50–57, Nov./Dec. 2011.
[12] J. F. Lin, "Low-power pulse-triggered flip-flop design based on a signal feed-through," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 1, pp. 181–185, Jan. 2014.
[13] H. Lee, S. Paik, and Y. Shin, "Pulse width allocation and clock skew scheduling: Optimizing sequential circuits based on pulsed latches," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 3, pp. 355–366, Mar. 2010.
[14] S. Paik, L.-E. Yu, and Y. Shin, "Statistical time borrowing for pulsed-latch circuit designs," in *Proc. 15th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2010, pp. 675–680.
[15] T. Baumann, D. Schmitt-Landsiedel, and C. Pacha, "Architectural assessment of design techniques to improve speed and robustness in embedded microprocessors," in *Proc. 46th Annu. Design Autom. Conf.*, Jul. 2009, pp. 947–950.
[16] Y. Zhang and B. H. Calhoun, "Hold time closure for subthreshold circuits using a two-phase, latch based timing method," in *Proc. IEEE SOI-3D-Subthreshold Microelectron. Technol. Unified Conf. (SS)*, Oct. 2013, pp. 1–2.
[17] H. Cheng, X. Li, Y. Gu, and P. A. Beerel, "Converting flip-flop to clock-gated 3-phase latch-based designs using graph-based retiming," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 4, pp. 979–992, Apr. 2022.
[18] K. Yoshikawa, K. Kanamaru, S. Inui, Y. Hagihara, Y. Nakamura, and T. Yoshimura, "Timing optimization by replacing flip-flops to latches," in *Proc. Asia South Pacific Design Autom. Conf.*, Jan. 2004, pp. 186–191.
[19] K. Y. Chung and S. K. Gupta, "Design and test of latch-based circuits to maximize performance, yield, and delay test quality," in *Proc. IEEE Int. Test Conf.*, Nov. 2010, pp. 1–10.
[20] J.-H.-R. Jiang and R. K. Brayton, "Retiming and resynthesis: A complexity perspective," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 25, no. 12, pp. 2674–2686, Dec. 2006.
[21] C. E. Leiserson and J. B. Saxe, "Retiming synchronous circuitry," *Algorithmica*, vol. 6, nos. 1–6, pp. 5–35, Jun. 1991.
[22] (2009). *Optimization With Pulp*. [Online]. Available: https://coin-or.github.io/pulp/
[23] MIT-LL. (2021). *Common Evaluation Platform (CEP)*. [Online]. Available: https://github.com/mit-ll/CEP.git
[24] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *Proc. ISCAS*, May 1989, pp. 1929–1934.
[25] F. Corno, M. S. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," *IEEE Des. Test. Comput.*, vol. 17, no. 3, pp. 44–53, Sep. 2000.

FILIPPO MINNELLA received the master's degree from Politecnico di Torino, in 2018, where he is currently pursuing the Ph.D. degree. He was with STMicroelectronics Automotive Group as an IC Designer focusing on digital circuits for mixed-signal devices and developing different SoC solutions. His main research interests include digital circuits optimization, EDA, and HLS.

JORDI CORTADELLA (Fellow, IEEE) received the Ph.D. degree in computer science from Universitat Politècnica de Catalunya, Barcelona, Spain, in 1987. He is a Professor with the Computer Science Department, Universitat Politècnica de Catalunya. His current research interests include formal methods and computer-aided design of VLSI systems, with a special emphasis on asynchronous circuits, concurrent systems, and logic synthesis. He is a member of Academia Europaea. He received the Best Paper Awards at the International Symposium on Advanced Research in Asynchronous Circuits and Systems, in 2004 and 2016, the Design Automation Conference, in 2004, and the International Conference on Application of Concurrency to System Design, in 2009. He has served on the technical committees of several international conferences in the field of design automation and concurrent systems.

MARIO R. CASU (Senior Member, IEEE) received the Ph.D. degree in electronics and communications engineering from Politecnico di Torino, Torino, Italy, in 2001. He is currently an Associate Professor with Politecnico di Torino. His research interests include systems-on-chip (SoC) with specialized accelerators, system-level design and design methodology for FPGAs and ASICs, and embedded machine learning. He is also interested in the design of circuits, systems, and platforms for industrial applications, such as biomedical, automotive, and food. His past work focused on the latency-insensitive design of SoC and networks-on-chip. He regularly serves on the Technical Program Committee for international conferences, such as DAC, ICCAD, and DATE.

MIHAI T. LAZARESCU (Senior Member, IEEE) received the Ph.D. degree in electronics and communications from Politecnico di Torino, Italy, in 1998. He is currently an Assistant Professor with Politecnico di Torino. He was a Senior Engineer with Cadence Design Systems and founded several startups. He has coauthored over 60 scientific publications, four books, and international patents. His research interests include design tools for the WSN/IoT platforms, ubiquitous environmental sensing, efficient neural networks, indoor human localization, edge and the leaf IoT data processing, and high-level HW/SW co-design and synthesis.

LUCIANO LAVAGNO (Senior Member, IEEE) received the Ph.D. degree in electrical engineering and computer science from UC Berkeley, in 1992. He was an Architect with POLIS HW/SW Co-Design Tool. From 2003 to 2014, he was an Architect with Cadence CtoSilicon High-Level Synthesis Tool. Since 1993, he has been a Professor with Politecnico di Torino, Italy. He has coauthored four books and over 200 scientific papers. His research interests include the synthesis of asynchronous circuits, HW/SW co-design, high-level synthesis, and design tools for wireless sensor networks.

• • •

Open Access funding provided by 'Politecnico di Torino' within the CRUI CARE Agreement