Mix & Latch: Comparison With State-of-the-Art Retiming On a RISC-V Benchmark

Mix & Latch: Comparison with State-of-the-Art Retiming on a RISC-V Benchmark

Lorenzo Lagostina, Filippo Minnella, Jordi Cortadella, Mario R. Casu, Mihai T. Lazarescu, and Luciano Lavagno

***Abstract*—Flip-flops (FFs) are the most commonly used sequential elements in synchronous circuits, but their timing requirements limit the operating frequency. Borrowing time with a latch-based approach can increase operating frequency, but traditional back-end optimization tools struggle to manage hold time requirements. The *Mix & Latch* technique achieves higher frequencies and often lower area than commercial state-of-the-art retiming by exploiting four types of synchronous sequential gates, namely positive and negative edge-triggered FFs and positive and negative transparent latches, all using a single clock tree. In this paper we first significantly accelerate the *Mix & Latch* flow convergence with respect to past work, by using a post-synthesis-based timing analysis that eliminates the first placement and routing needed for post-layout timing analysis. Then, by adding tolerance margins to the timing model, the pessimism is reduced to improve both convergence speed and maximum frequency. Finally, we reduce the complexity of the problem by applying the methodology only to the sequential elements belonging to critical paths. The effectiveness of *Mix & Latch* is then demonstrated on a *RISC-V* processor core from the *Pulp* platform using $28\,\mathrm{nm}$ *CMOS FDSOI* technology. The results are compared to both the original *Mix & Latch* flow and a retiming performed with a state-of-the-art tool, showing a $25\,\%$ frequency improvement over the original flow and $7.5\,\%$ over the retiming flow. Compared to the retiming flow, we achieve comparable or lower power and area, while preserving the original registers and allowing logic equivalence checking.**

***Index Terms*—Integrated circuit synthesis, Design automation, Sequential circuits, Latches, Flip-flops, Risc-V.**

I. Introduction

Sequential circuits use *either* flip-flops (FFs) *or* latches for data storage. Latches can be used in fault-tolerant applications [1], can operate at lower supply voltages, reduce power consumption [2], [3], [4], and increase operating frequency [5], [6]. However, more complex timing constraints limit their support by commercial flows and their use in industrial designs.

One way to increase the operating frequency of integrated circuits is to use latches to exploit time borrowing. However, time borrowing is traditionally limited by latch timing constraints, especially hold times, which are complex to satisfy. Previous work has used narrow clock pulses, which are difficult to distribute, or buffers, which increase area, to address the problem [7]. To fully take advantage of latches, the *Mix & Latch* method published in [8] solves the hold time constraints with the help of negative transparent latches (NTLs) that act as retention barriers. The resulting sequential circuit implementation uses positive transparent latches (PTLs), NTLs, positive-edge-triggered flops (PETFs) and negative-edge-triggered flops (NETFs) as primary storage elements, all driven by a *single clock tree*, as explained in Section II.

Although *retiming* can improve the operating frequency during logic synthesis [9] (and is provided by most electronic design automation commercial tools), it significantly hampers design verification [10] because it changes the locations of registers in the netlist, drastically increasing the complexity of equivalence checking with state-of-the-art commercial tools. As explained later, *Mix & Latch* does not suffer from this limitation, since it ensures that for each original FF there is a corresponding final register, which may not be a FF, but which can act as an anchor point for checking combinational equivalence.

The original *Mix & Latch* [8] improves performance by:

1) using PTLs over FF-based implementations allowing time borrowing,
2) adding NTLs to optimally resolve the resulting hold time violations,
3) replacing adjacent NTL-PTL pairs with PETF & NETF.

In this work we aim to further increase performance with respect to [8], reaching a level that is at least comparable to retiming and avoiding resource increase.

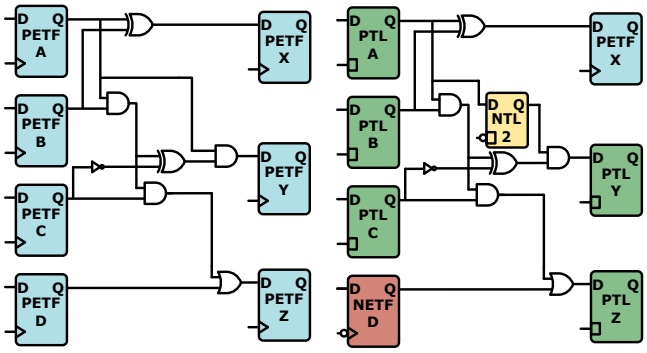
Our main contributions to the state-of-the-art are:

1) Speed up the original *Mix & Latch* flow by avoiding one place and route (P&R) iteration.
2) Relax the timing analysis both in terms of setup constraints, which control the performance of the circuit after NTL insertion, and of how to select which hold violations the NTL insertion will solve.
3) Propose a flow considering only setup critical paths for PTL insertion.
4) Evaluate *Mix & Latch* effectiveness by comparing it to state-of-the-art commercial retiming performed after logic synthesis. We use as benchmark a *Zeroriscy* [11] RISC-V core, achieving a $25\,\%$ frequency improvement over the original flow, with comparable area and power at the highest frequency result achieved by the original flow. Additionally, we achieve a $7.5\,\%$ frequency increase over retiming, while reducing on average area and power by $12.63\,\%$ and $5.53\,\%$ respectively.

II. Mix & Latch Optimization Flow: Original vs. New

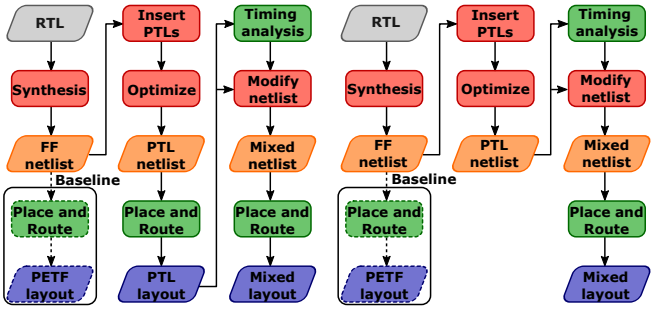
Fig. 1 shows the *Mix & Latch* methodology using a simplified example. Starting from a FF-based netlist (Fig. 1a), all registers are replaced by PTLs to fully exploit time borrowing, but at the risk of hold time violations. To solve them, *Mix & Latch* inserts NTLs on the short paths that are at risk. The short paths can be categorized as either register-to-register connections or launched/captured by an I/O pin. Such latches act as retention barriers, delaying any signal that travels through a short path.

Manuscript created August, 2023; L. Lagostina, F. Minnella, M. R. Casu, M. T. Lazarescu, and L. Lavagno are with the Department of Electronics and Telecommunications (DET), Politecnico di Torino, Torino, I-10129, Italy (email: luciano.lavagno@polito.it); J. Cortadella is with the Department of Computer Science, Universitat Politècnica de Catalunya, Barcelona, S-08034, Spain (email: jordi.cortadella@upc.edu).



(a) Generic PETF-based circuit (b) Resource merging

Fig. 1. Example of *Mix & Latch* optimization on a PETFs-based circuit (Fig. 1a). Merge sequences of complementary latches: PTL-to-NTL into NETF, NTL-to-PTL into PETF (Fig. 1b). Combinational gates have unit delay only for ease of explanation. We consider as short paths those that cross one gate.



(a) Original flow (b) New flow

Fig. 2. Comparison between the original (Fig. 2a) and the new post-synthesis (Fig. 2b) implementation flow. The new flow performs timing information extraction and netlist manipulation after the PTL-based netlist synthesis.

Choosing the best location for these latches is not trivial. In fact, if placed incorrectly, these NTLs lead to setup violations. Additionally, the latches can be merged if there is a direct PTL-to-NTL or NTL-to-PTL connection, resulting in either a PETF or a NETF, as shown in Fig. 1b.

Mix & Latch uses an integer linear programming (ILP) model to compute a cut from the PTL post-layout circuit graph determining the best NTLs position. The ILP model aims to minimize the new sequential elements (NTLs) added to the circuit, resolve hold violations, and avoid performance degradation.

Three timing constraints limit the potential timing degradation produced by the insertion of NTLs in critical paths and help satisfy the setup constraints. The cost function optimizes the area overhead. As shown in [8], the ILP model is similar to a max-flow formulation, and the runtime has proven to be manageable, the implementation flow incurs a maximum $7\,\%$ overhead, even for large designs, which means circuits with $10\,\mathrm{k}$ FFs.

Fig. 2 shows a comparison between the original flow and our new proposal for the *Mix & Latch* methodology.

To overcome the limitations of the original *Mix & Latch* flow, we introduce three innovations: (1) pessimism reduction, (2) post-synthesis extraction of timing information, (3) limitation of the analysis to only setup critical flip-flops. The rest of this section discusses the proposed improvements.

A. Post Synthesis Flow

The original *Mix & Latch* flow has typically long execution times because it requires two P&R steps and derives the timing graph (TG) taking into account the delay contributions from the interconnections. However, since it only *estimates* the best positions for NTLs, it cannot guarantee that the solution will produce the desired results, since cell placement and routing is left to the P&R tool, and the resulting netlist changes cannot be handled by the tool in incremental mode (also known as ECO mode). To address both problems, a so-called *post-synthesis* flow is introduced, where the TG and cut computations are performed on the netlist after the synthesis step, as shown in Fig. 2, instead of after the P&R of the PTL-based netlist.

B. Reduce Pessimism

As shown in Fig. 2a, the original *Mix & Latch* methodology uses post-P&R information to estimate the effect of the NTL insertion. However, since the netlist changes caused by NTL and NETF are too large to be handled by incremental implementation, the estimate of the impact of their insertion performed by *Mix & Latch* is necessarily imprecise.

For setup time, some promising solutions may be discarded due to small negative values of the estimated setup slack after NTL insertion, ESS_p, which poses a hard exclusive constraint to the solver and instead may be fixed with P&R. For hold time, it may not be necessary to resolve all violations via NTLs, as small ones can be resolved by the P&R tool by inserting buffers and/or resizing cells. To address these issues, two parameters are introduced to reduce the pessimism of the algorithm:

- Max setup derate MSD;
- Max hold derate MHD.

MSD is multiplied by the clock period T and added at the end of [8, Algorithm 1], where the final line becomes:

$$ESS_p \leftarrow ESS_p + T \times MSD. \tag{1}$$

In this way, small negative values of ESS_p become positive, thus avoiding to discard a potentially viable solution. As discussed in Section III, small values of this parameter (e.g., $10\,\%$ of T) are sufficient to achieve better performance.

While MSD tries to increase the number of possible solutions, MHD reduces the number of paths included in the short-path subgraph, leaving the solution of some hold violations to the P&R tool. The results show that small values of this parameter (e.g., also $10\,\%$ of T) also help to achieve better performance, while too high values would lead to excessive hold violations that cannot be fixed in the final layout.

C. Worst-Path Flow

Another strategy for reducing the complexity of the optimization problem is to only take into account the flip-flops that are endpoints of the setup critical paths rather than converting all of them to PTLs. This is called *worst-path* flow and use the Worst Setup Slack (WSS) parameter. Only flip-flops with

setup slack less than WSS at the input data pin are taken into account. Experimental results show that the optimal value for this parameter can change based on the target frequency.

D. Formal verification

[12] shows that keeping each register in its original position, and possibly adding registers that will be ignored in formal verification, as done in *Mix & Latch*, simplifies formal verification. The equivalence of circuit behavior involves a combination of Logic Equivalence Checking (LEC) and static timing analysis (STA). STA guarantees that modifying the characteristics of registers does not introduce data sampling errors compared to the original FF-based netlist, and is fully discussed in [8]. LEC ensures that the introduction of retention barriers (NTLs) in the circuit does not alter the behavior of the combinational logic. During LEC, the NTLs operate in transparent mode, setting their input clock to 0. Differently from *retiming*, this methodology does not require annotations from the synthesis step.

III. Experimental Results

A. Test Setup

We selected a *Zero-riscy* core [11] to be implemented using the above flows because RISC-V cores have become increasingly popular in recent years, and because processors contain both acyclic and cyclic paths, e.g., in arithmetic units and FSMs respectively. This variety of subcircuit topologies helps us to test the *Mix & Latch* methodology under stringent conditions, since the original work [8], evaluated on a broad set of circuits, showed that it is most effective on acyclic circuits. To perform the power analysis with realistic switching activity information, the standard delay format simulation step is performed with backannotated delays obtained while running advanced encryption standard (AES), 2D convolution and matrix multiplication algorithms on the RISC-V core. At the end of the test, the flow compares the values stored in the data memory with those obtained during a reference RTL simulation to ensure the correctness of the result. This increases confidence in the correctness of the implementation because it validates the timing constraints added to the backend flow.

The RISC-V core is synthesized with Synopsys Design Compiler, which also performs the retiming optimization used for comparison. Physical synthesis is then performed using Cadence Innovus. We use a $28\,\mathrm{nm}$ *CMOS FD-SOI* technology.

From a sweep of their value from $0\,\%$ to $20\,\%$, the best value obtained for both MSD and MHD (discussed in Section II-B) is $10\,\%$ of the clock period for the new *Mix & Latch* flows.

Since the cell library does not provide NETFs cells, we used PETFs cells with an inverter on the clock pin. Thus, replacing a PTL-NTL pair with a NETF to increase frequency penalizes the design density due to the additional inverters, and we disabled it during the experiments. The ILP takes approximately $70\,\mathrm{s}$ to complete, each layout iteration approximately $50\,\mathrm{min}$. The speedup of the *post-synthesis* flow is $30\,\%$ with respect to [8].

The *worst path* variation is applied on top of the *post-synthesis* flow. To adjust the WSS threshold for the worst-path flow, the setup slack of the critical path is used as a starting point and a sweep is performed in $50\,\mathrm{ps}$ steps until the clock period value. Then is selected the WSS value which delivers the best area and power performance after achieving STA closure. Automatic tuning of this parameter is left for future work.

TABLE I
Performance comparison of implementation flows

Implementation flow	$T_{\min}$ (ns)	$f_{\max}$ (MHz)	f_{gain}
FF without retiming (baseline)	3.00	333.3	$1.0\times$
Mix & Latch original	2.50	400.0	$1.20\times$
FF with retiming	2.15	465.0	$1.40\times$
Mix & Latch post-syn	2.00	500.0	$1.50\times$
Mix & Latch worst-paths	2.00	500.0	$1.50\times$

B. Performance, Area, and Power Analysis

For each flow, a clock period constraint sweep is performed to find the maximum operating frequency. Table I shows the performance results for minimum clock period $T_{\min}$, maximum frequency $f_{\max}$ and relative frequency gain f_{gain} compared to the baseline FF-based implementation.

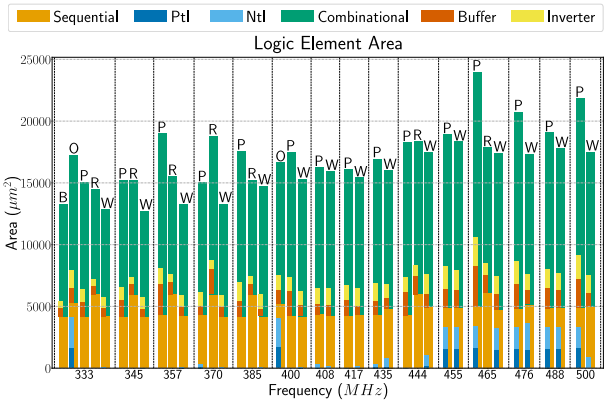
Compared to baseline implementation, both the post-syn flow and the worst-path version of *Mix & Latch* flow improve the frequency over classical retiming, showing the same performance boost. This result, obtained on a more complex benchmark than [8], is very promising for the *Mix & Latch* methodology, implying that it may become a viable alternative to logic retiming in industrial flows.

Note that the original *Mix & Latch* flow applied to this design failed, because it added too many NTLs, which eventually caused timing violations in the mixed layout. The $400\,\mathrm{MHz}$ clock frequency was achieved only by using the MSD and MHD parameters introduced in this paper. This suggests that the choice of best values for the *Mix & Latch* parameters (like many other parameters in modern physical implementation flows) may vary with design characteristics. For example, the Zero-riscy RISC-V core may have more short paths than the previously considered benchmarks, and relaxing the timing constraints of *Mix & Latch* seems an effective strategy to address this issue.

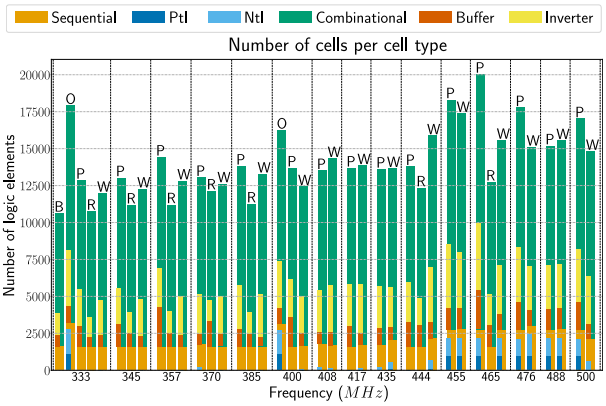
Fig. 3 and Fig. 4 plot power, area, and cell usage for each implementation flow versus clock frequency, highlighting their components. Missing points in the plots are caused by failed STA at the end of the selected flow. Note that the original *Mix & Latch* flow struggles the most to achieve timing closure, while the *post-synthesis* and *worst path* flows both provide a feasible solution for every target frequency.

In general, designs optimized using both the original and *post-synthesis Mix & Latch* flows consume more power than the retimed design, while their area is comparable or smaller, except at the highest frequencies, as shown in Fig. 3a. The *worst path* flow instead manages to deliver the smallest area occupation compared to the other flows for each target frequency. Moreover, it also achieves smaller power consumption, apart from the highest retiming frequencies.

The area and power results show how the original *Mix & Latch* flow is suboptimal with respect to the newest improvements. This is probably because the *Mix & Latch* algorithm

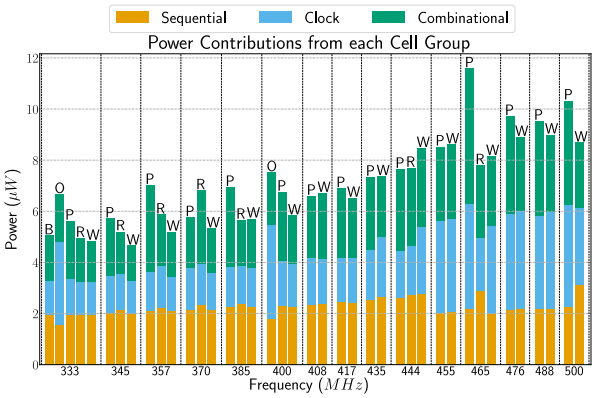


(a) Area of different logic cell types

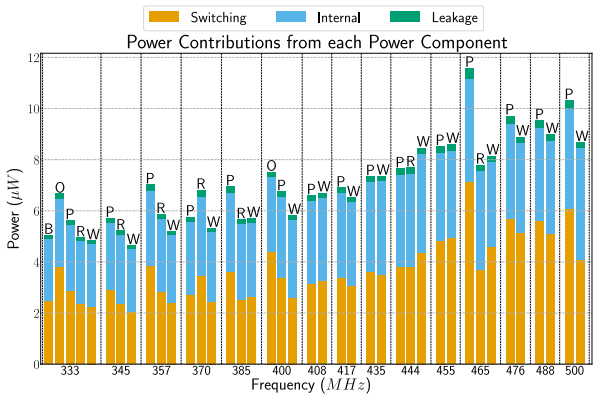


(b) Usage of different logic cell types

Fig. 3. Total area at different frequencies for baseline (B), original *Mix & Latch* flow (O), post-synthesis *Mix & Latch* (P), retiming (R), worst path (W), split into sequential and combinational contributions highlighting the fractions related to PTLs, NTLs, buffers, and inverters (Fig. 3a) and number of logic elements (Fig. 3b).



(a) Contribution of each cell group to the power consumption



(b) Internal, switching, and leakage power consumption

Fig. 4. Power consumption for original *Mix & Latch* (O), post-synthesis *Mix & Latch* (P), and retiming (R) at different operating frequencies, split by cell group (Fig. 4a) and by type of power (Fig. 4b).

has trouble predicting the choices the P&R tool makes in the second layout step. Instead, the *post-synthesis* implementation delivers on average a power and area overhead of $11.10\,\%$ and $5.33\,\%$, respectively, compared to the retiming flow. Finally, the *worst path* implementation achieves a power reduction of $5.53\,\%$ compared to retiming along with an area reduction of $12.63\,\%$. Therefore, it is possible to use either the *post-synthesis* or *worst path* flow based on power, performance, and area (PPA) vs. design time constraints, since the first flow provides a viable solution at the first iteration, while the second requires some additional steps to fine-tune the WSS threshold.

C. Logic Cell Utilization

Comparing retiming with *Mix & Latch* results in Fig. 3a, it can be observed how using latches instead of FFs reduces the area occupied by sequential elements, even when the number of sequential cells is nearly doubled (see Fig. 3b). This is due to both latches being smaller than FFs and retiming resizing cells to improve performance, while we use the smallest PTL and NTL cells to further reduce area usage.

Retiming also seems to use more numerous and larger buffers compared to *post-syn* and *worst path Mix & Latch*, probably due to resizing like for FFs. Instead, the introduction of even a very small number of latches leads to a significant increase in inverter cells with respect to retiming. This result is explained by the stricter clock tree constraints used during clock tree synthesis when using both sequential cell types with different clock polarities. In our case, the P&R tool decided to instantiate more inverter cells, as shown in Fig. 3b, leading to a corresponding increase in inverter area in Fig. 3a.

It can also be observed how the results of the *Mix & Latch* flows can vary with timing constraints. For target frequencies from $455\,\mathrm{MHz}$ to $488\,\mathrm{MHz}$, the *Mix & Latch* algorithms choose solutions in which $62.5\,\%$ of the original registers are kept as latches, while in most of the other observed cases, at *both lower and higher target frequencies*, this percentage is less than $5\,\%$. Since the final results highly depend on the hyper-parameters of the *Mix & Latch* algorithm, further research is required to obtain better predictions and avoid the manual tuning of the hyperparameters.

D. Power Contributions

Fig. 4 summarizes the power consumption estimates extracted from the back-annotated simulation. Fig. 4b shows the different power components: (1) switching power, consumed by charging and discharging the interconnect and load capacitances, (2) internal power, consumed by charging and discharging the internal gate capacitance and by short-circuit currents,

and (3) leakage power. Fig. 4a shows power contributions broken down by type of logic elements: (1) clock tree cells, (2) combinational logic, and (3) sequential elements.

Switching power is the main component in Fig. 4b that determines the power overhead of the original and *post-synthesis Mix & Latch* implementations. This can be explained by the fact that PTLs and NTLs in the final design allow more glitches to propagate along logic paths. The *worst path* flow, on the other hand, manages to successfully reduce switching power relative to the other *Mix & Latch* flows, and also reduces this component relative to retiming at the lower target frequencies.

Another major contribution is clock tree elements, especially when most of the registers are ultimately kept as latches, as in the original *Mix & Latch* flow or the other two flows in the range from $455\,\mathrm{MHz}$ to $488\,\mathrm{MHz}$, as discussed in Section III-C. It is also possible to assess a positive correlation between the increase in inverter cells and clock power, further indicating how the *Mix & Latch* methodology increases the number of elements on the clock tree. The analysis of the necessary modifications to the clock gating for *Mix & Latch* is left to future work.

While some configurations of *Mix & Latch* significantly increase the clock power, these same configurations reduce internal and sequential power, both of which can be attributed to the reduction in the size of the sequential elements.

Static power increases for the original and *post-synthesis Mix & Latch* flows relative to the retimed implementation, while the *worst path* variant provides the lowest leakage power in most of its configurations.

Overall, the main term of the power graphs is the switching component, since Fig. 4b shows how the sum of all power components follows the same trend as the switching one. This suggests that by addressing glitch propagation and reducing the number of clock tree elements, it should be possible to minimize the power overhead of the *post-synthesis* implementation and achieve comparable values to the retiming implementation, while avoiding the multiple iterations required for the *worst path* flow.

E. Mix & Latch With Retiming

Finally, we tried the new *Mix & Latch* flow *in conjunction* with retiming to further explore its potential. The P&R step only succeeds if retiming is performed either before TG extraction or after mixed netlist generation. Applying *Mix & Latch* to a retimed FF-based netlist or retiming after the PETF-to-PTL step results in a final layout that does not satisfy the timing constraints. In both cases, if P&R is successful, *there is no performance gain over the original* Mix & Latch *flow*, and in fact power and area usage increase compared to the results in Fig. 3 and Fig. 4. Although these results may be design dependent, they seem to imply that *the proposed* Mix & Latch *flow and retiming exploit essentially the same degrees of freedom in optimizing performance without changing the combinational logic*, and that no substantial gains can be achieved by combining them.

Thus, for the analyzed design, *Mix & Latch* is superior to retiming in terms of both maximum achievable performance and *compatibility with combinational equivalence checking*.

IV. Conclusions

In this paper, we presented a variation of the original *Mix & Latch* flow that aims to reduce runtime and increase performance to catch up with conventional retiming methods.

As an example benchmark, we tested the updated *Mix & Latch* flow on a *Zeroriscy* RISC-V core.

First, we introduced tolerances in the timing analysis to give *Mix & Latch* more freedom to choose the NTL positions. The PTL netlist layout was then avoided to reduce execution time by performing the timing analysis required by *Mix & Latch* to determine the position of NTLs in the netlist earlier, in the post-synthesis phase.

The results show a $25\,\%$ clock frequency improvement over the original flow and a $7.5\,\%$ over retiming, with an average $5.53\,\%$ lower power consumption and $12.63\,\%$ lower area occupation. The PPA improvements prove that the *Mix & Latch* methodology can challenge traditional retiming-based techniques while avoiding the drawbacks that make the latter harder to use in industrial design flows, i.e., by allowing combinational equivalence checking to be used throughout the flow.

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 Journal of Solid-State Circuits*, vol. 48, no. 1, pp. 66–81, 2013.

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

[3] N. A. N. Hassan, A. B. Abd Manaf, and L. C. Ming, "Optimization of circuitry for power and area efficiency by using combination between latch and register," in *2011 IEEE International Conference on Computer Applications and Industrial Electronics (ICCAIE)*, 2011, pp. 240–244.

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

[5] A. P. Hurst and R. K. Brayton, "The Advantages of Latch-Based Design Under Process Variation," in *Proceedings of the IWLS*, 2006.

[6] B. Taskin and I. Kourtev, "Time borrowing and clock skew scheduling effects on multi-phase level-sensitive circuits," in *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, vol. 2, 2004, pp. II–617.

[7] N. Shenoy, R. Brayton, and A. Sangiovanni-Vincentelli, "Minimum padding to satisfy short path constraints," in *Proceedings of 1993 International Conference on Computer Aided Design (ICCAD)*, 1993, pp. 156–161.

[8] F. Minnella, J. Cortadella, M. R. Casu, M. T. Lazarescu, and L. Lavagno, "Mix & Latch: An Optimization Flow for High-Performance Designs With Single-Clock Mixed-Polarity Latches and Flip-Flops," *IEEE Access*, vol. 11, pp. 35 830–35 840, 2023.

[9] C. E. Leiserson and J. B. Saxe, *Retiming Synchronous Circuitry*, 1988.

[10] 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 *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2018, pp. 203–208.

[11] P. Davide Schiavone, F. Conti, D. Rossi, M. Gautschi, A. Pullini, E. Flamand, and L. Benini, "Slow and steady wins the race? A comparison of ultra-low-power RISC-V cores for Internet-of-Things applications," in *2017 27th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2017, pp. 1–8.

[12] K. Yoshikawa, K. Kanamaru, S. Inui, Y. Hagihara, Y. Nakamura, and T. Yoshimura, "Timing optimization by replacing flip-flops to latches," in *ASP-DAC 2004: Asia and South Pacific Design Automation Conference 2004 (IEEE Cat. No.04EX753)*, 2004, pp. 186–191.