Saving Power by Converting Flip-Flop to 3-Phase Latch-Based Designs

Huimei Cheng
huimeich@usc.edu

Xi Li
xli497@usc.edu

Yichen Gu
yichengu@usc.edu

Peter A. Beerel
pabeerel@usc.edu

Ming Hsieh Department of Electrical and Computer Engineering
University of Southern California, Los Angeles, CA

***Abstract*—Latches are smaller and lower power than flip-flops (FFs) and are typically used in a time-borrowing master-slave configuration. This paper presents an automatic flow for converting arbitrarily-complex single-clock-domain FF-based RTL designs to efficient 3-phase latch-based designs with reduced number of required latches, saving both register and clock-tree power. Post place-and-route results demonstrate that our 3-phase latch-based designs save an average of 15.5% and 18.5% power on a variety of ISCAS, CEP, and CPU benchmark circuits, compared to their more traditional FF and master-slave based alternatives.**

I. Introduction

The growing use of portable/wireless electronic systems and Internet-of-Things (IoT) applications motivates the desire for energy-efficiency in today's very large scale integration (VLSI) circuits. One of two devices: edge-triggered flip-flops (FFs) or level-sensitive latches are typically used as synchronization and storage. Compared to FFs, latches have the advantages of time borrowing, skew and jitter tolerance, smaller cell area, and lower capacitance. Latch-based designs can thus consume lower power and area than FF-based designs [1]–[3], particularly when process variation is considered [4]. Latch-based designs are also critical for architecturally-agnostic timing resilient designs [5], [6] which can remove unnecessary margins associated with PVT variations and make near-threshold computing more practical.

As an intermediate between latch and flip-flop based designs, pulsed-latch schemes have also been proposed [7], [8]. These rely on an edge-triggered pulse generator to provide a short transparency window to all latches. To minimize energy overhead, multi-bit pulsed-latch schemes have been proposed to share pulse generators among several latch cells [9]. Pulsed-latches, however, must be used carefully because they are subject to hold problems and pulse width variations that are challenging to predict, control, and mitigate (see e.g., [10]).

A basic challenge to adopting any form of latch-based design is that most RTL specifications are designed using edge sensitive FFs. Approaches to automatically converting a FF- to latch-based design are thus attractive. Most conversion flows convert the FF-based designs into pulsed-latch designs [11] or two-phase latch-based designs controlled by either master-slave clocks [12] or bundled-data asynchronous controllers [13].

Optimization of latch-based designs has also been given some attention in the literature. For example, [2] explores using a mix of master-slave latches and FFs/pulsed-latches. Others take advantage of the time borrowing to boost performance and/or reduce area and power consumption [2], [12]. Moreover, retiming algorithms of timing-resilient latch-based designs have been developed that consider not only the number of latches required but also the impact of the amount of needed error-detecting logic [14].

Whereas two-phase designs are inherently more robust than pulsed-latch designs, we argue they can be overly restrictive and that multi-phase latch-based designs [15] can sometimes be an attractive alternative.

The key contribution of this paper is to demonstrate that a FF-based design can be automatically converted into a power efficient multi-phase design. In particular, we convert a FF-based to 3-phase latch-based design using a novel Integer Linear Program (ILP) that minimizes latches and apply retiming and clock-gating to optimize the design. Our post place-and-route experimental results show an overall average of 15.5% and 18.5% reduction in power compared to FF-based and master-slave latch-based designs on ISCAS89 circuits [16], CEP submodules [17], and three CPU designs (i.e. a 3-stage MIPS CPU Plasma [18], a RISC-V Rocket Core [19], and an ARM Cortex-M0 core [20]) .

This paper is organized as follows. Section II introduces background on multi-phase latch-based designs. The motivation targeting a 3-phase clocking is discussed in Section III. Section IV introduces our ILP-based conversion algorithm and Section V presents the experimental results based on a broad range of designs. Finally, some conclusions are drawn in Section VI.

II. Background

A. Multi-Phase Latch-Based Designs

The Sakallah, Mudge, and Okulotun (SMO) model [15] defines an optimal framework for multi-phase latch-based designs. It defines a k-phase clock as a collection of k periodic signals with a common cycle time and associated timing constraints, called the General System Timing Constraints (GSTC). The phases (p_1, p_2, ... p_k) are ordered in a global time reference: $e_{i-1} \leq e_i$; $e_k = T_c$, where e_i is the closing

978-3-9819263-4-7/DATE20/©2020 EDAA

time of phase p_i. E_{ij} is the forward phase shift from phase p_i to phase p_j defined below.

$$E_{ij} = \begin{cases} (e_j - e_i), & i < j \\ (T_c + e_j - e_i), & i \geq j \end{cases} \quad (1)$$

Then, the worst-case setup and hold constraints for each phase is defined as follows.

$$\begin{aligned} &\text{Hold: } H_i \leq d_j + \delta_j + \delta_{ji} - E_{p_j p_i} \\ &\text{Setup: } T_c - S_i \geq D_j + \Delta_j + \Delta_{ji} - E_{p_j p_i} \end{aligned} \quad (2)$$

Here, H_i and S_i stands for the hold and setup time of the i^{th} latch. The shortest (longest) path delay from the j^{th} latch to the i^{th} latch is denoted as δ_{ji} (Δ_{ji}) and the minimal (maximal) delay value of the j^{th} latch is δ_j (Δ_j). d_j (D_j) represents the earliest (latest) signal departure time, i.e., the amount of time after the last e_j that the next data starts to propagate through the j^{th} latch [15]. T_c denotes the cycle time.

III. Motivation and Goal

This paper's goal is to convert a FF-based to latch-based design minimizing the number of latches based on a reasonable set of constraints.

A. Conversion Constraints

The first two constraints we adopt are designed to make the application (e.g. reset states, verification, and testing) of latch-based designs easier. The third constraint ensures the same throughput, meaning an fair comparison to FF-based designs.

C1: the original position of all FFs must be latched;
C2: neighboring latches, connected by combinational logic, must not be simultaneously transparent;
C3: the converted latch-based design must have the same throughput as the FF-based design assuming the combinational logic is already critical.

B. Special Case of Linear Pipelines

It is interesting to consider the special case of a linear pipeline because it has no FFs with combinational feedback that must be considered. In Fig. 1, a linear FF-based pipeline (a) can be converted to a 3-phase latch-based pipeline (b). The conversion adds exactly one extra latch stage for every other original pipeline stage, which can be shown to be the minimum number of extra latches possible while still meeting all the constraints described in Subsection III-A [21]. This paper explores the 3-phase opportunities in more general non-linear pipelines.

IV. Conversion Algorithm

Our conversion approach is to automatically decompose the FFs into two groups, ones that will be converted into a single latch and ones that will be converted to back-to-back connected latches. The first group of FFs are converted to a single latch and assigned to clock phase p_1. Remaining FFs are in the second group and converted to latches clocked by either p_1 or p_3. For this group, an additional latch clocked by p_2 is inserted at each latch's output to create a back-to-back configuration. This means that, by construction, there is no direct data path from p_3 to p_1 latches. Min delay related hold problems are avoided by allowing an FF to be assigned to phase p_1 and converted to a single latch only if none of its fanout FFs are also assigned to p_1.

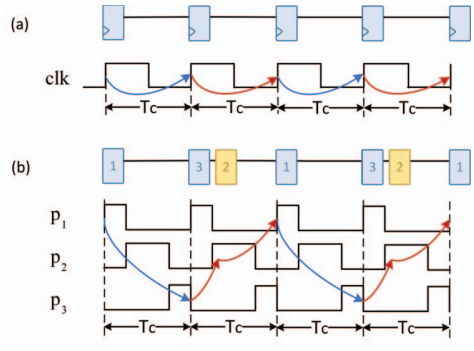

Fig. 1: Converting (a) a linear FF-based pipeline to (b) a 3-phase latch-based pipeline. The number labeled in a latch represents the phase that drives the latch. Latches driven by p_2 (in yellow) are inserted exactly one extra latch stage for every other original pipeline stage.

A. Integer Linear Programming (ILP)

Each FF is treated as a node u and its $FO(u)$ is the set of FFs that can be reached from the FF via only combinational logic. Every node u has two binary parameters, G and K. $G(u)$ decides which group of latches to assign the node u to, either the back-to-back latch group with $G(u)$ being 1, or the single-latch group where $G(u)$ is 0. $K(u)$ determines the node u's clock phase, 1 implies u is clocked by p_1 and 0 implies u is clocked by p_3. All inserted latches are driven by p_2. Our ILP automatically performs this assignment minimizing the number of back-to-back latches as follows:

$$Minimize \sum_u G(u)$$

Subject to:

$$\forall u \in V: \quad G(u) = \begin{cases} 1, & K(u) = 0, \\ 1, & K(u) = 1 \wedge \exists v \in FO(u)\ K(v) = 1 \\ 0, & otherwise \end{cases}$$

$$K(u) = \begin{cases} 1, & \forall u \in \textit{PI} \\ \{0, 1\}, & \forall u \in V \end{cases}$$

Here *PI* stands for the set of all primary input ports and set V contains all FFs in the circuit. To provide consistency to the interface of the design, we assign all primary input ports (*PI*s) as if they were clocked by p_1.

To make the ILP compatible with Gurobi [22], we convert the conditional equations into inequalities:

$$Minimize \sum_u G(u)$$

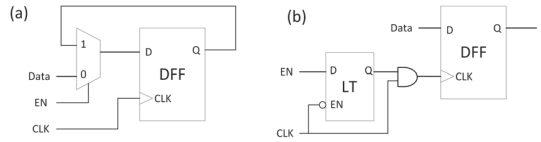

Fig. 2: Clock gating styles: (a) enabled clock and (b) gated clock

Subject to:

$$\begin{cases} G(u) + K(u) \geq 1 & \forall u \in V \\ G(u) \geq K(u) + K(v) - 1 & \forall u \in V, \forall v \in FO(u) \\ G(u) \geq K(v) & \forall u \in \textbf{\textit{PI}}, \forall v \in FO(u) \end{cases}$$

$$G(u), K(u) \in \{0, 1\}$$

The first constraint implies if $K(u)$ is 0 then inequality $G(u) \geq 1$ is satisfied, in other words, a p_3 latch is always in a back-to-back latch group. The second constraint makes sure to insert a latch for consecutive p_1 latches by forcing $G(u) \geq 1$ when both $K(u)$ and any of its fanout $K(v)$ are 1. Given the assumption of all *PI*s to be clocked by p_1, we can obtain the third constraint with $K(u) = 1$ from the second constraint.

B. The Design Flow

The ILP described in the last section is the core step in a design flow that supports FF-based to 3-phase latch-based design conversion. The first step of our design flow is to run standard synchronous synthesis on the given FF-based RTL design. Here we take care to enable clock gating to minimize the number of FFs with self-loops which would otherwise unduly constrain the optimization problem. To be specific, the gated clock, shown in Fig. 2(b), is set to be the preferred clock gating style, as compared to enabled clock illustrated in Fig. 2(a).

Using Python and TCL scripts that interface a leading commercial logic synthesis tool to the Gurobi Integer Linear Program solver [22], we then take the resulting FF-based design, identify the connections between FFs, and formulate the ILP described in Section IV-A. We run the ILP, and, using the results, create the equivalent 3-phase latch-based synchronous design by defining the three-phase clocks and connecting them to their associated latches. For each latch that is clock gated, we trace the clock signal back through the clock gating logic and replace the clock with $p1$ or $p3$. In the case of latches belonging to the same clock gating register bank but driven by different clock phases, the clock gating logic is duplicated and connected to the two clock phases separately.

We then apply retiming and clock gating strategies to the newly added latches, which will be discussed in Subsections IV-C and IV-D. The last step in the design flow is the physical design step which includes implementation of the three-phase clock trees.

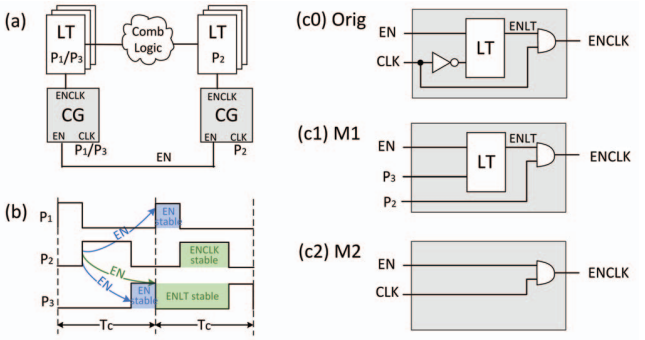

Fig. 3: (a) Clock gate p_2 latches whose leading latches share a common enable EN. (b) Timing diagram of EN. (c0) A typical CG cell. (c1) Modified CG for p_2 latches. (c2) Modified CG for $p1$/p_3 latches.

C. Modified Retiming

Retiming re-positions the added latches within the combinational logic minimizing area while satisfying all latch constraints. Unfortunately, many commercial tools have limited support for retiming latches. They do, however, have well-optimized support for the retiming of FFs. Using this fact, [23] proposed to retime latches by mapping it to a FF-based retiming problem. Given a synthesized design with clock period T_c, they replace each FF with two FFs and retime the entire design with a faster clock constraint of half the original period ($T_c/2$). After splitting the combinational logic, the FFs are converted into alternating transparent low and high latches.

In this paper, instead of halving the cycle, we keep the cycle time unchanged but use back-to-back FFs, where the first FF is controlled by clk and the second clocked by clk inverted (clkbar). Both clk and clkbar have the high pulse width to be half of the cycle time. The group that is converted to a single latch is replaced with a single FF, also controlled by clk. In other words, the 3-phase clocks are mapped to clk and clkbar, with phase p_1 and p_3 mapped to clk and p_2 tied to clkbar. We then retime the circuit by only allowing FFs tied to clkbar to move. This splits the combinational logic in the pipeline stages that require an extra latch into two with each part being able to operate at twice the frequency (cycle time $T_c/2$).

After the relocation of FFs clocked by clkbar, all FFs can be converted back to latches with their designated 3-phase assignments. Further optimization is then triggered to optimize the sizes of gates in the retimed latch-based design.

D. Clock Gating (CG)

A clock gate shuts off a branch of the clock tree when it is not needed to save dynamic power. In this section, we apply clock gating to the newly added p_2 latches. For a fair comparison, no extra clock gating is applied to latches clocked by p_1 or p_3. Our first means of clock gating p_2 latches is to gate those latches whose upstream latches are also gated. In particular, we identify every individual p_2 latch whose fan-in latches share a common enable signal EN. We then can clock gate the p_2 latch using a separate CG cell also controlled by

EN, later we simplify it as "p_2 CG", illustrated in Fig. 3(a). A typical CG cell, Fig. 3(c0) for example, would be composed of a latch, an inverter generating the inverted clock, and an AND gating the clock. While this conventional cell is generally applicable, we apply two modifications to reduce overhead that are specific to three-phase designs.

Our first modification (M1) is to remove the inverter and use p_3 rather than the inverted p_2 in a p_2 CG. As illustrated in Fig. 3(c1), a new pin p_3 is introduced to the p_2 CG, and the clock pin (CLK) is specifically tied to p_2. Interestingly, the timing constraints associated with clock gating are easily met. Recall that the enable signal EN that clock gates the p_2 latch also clock gates its upstream latches. As highlighted in blue in Fig. 3(b), the EN signal is guaranteed to be stable when the upstream latches open. This means the EN that controls a p_2 CG, highlighted in green, becomes valid before the rising edge of p_1 and it thus basically safe to be latched using p_3 (assuming only a small (if any) gap between p_1 rising and p_3 falling). The output of the latch in CG, labeled ENLT in Fig. 3(c1), is thus stable from the falling edge of p_3 until the next rising edge of p_3. This guarantees the stability of ENCK during the high period of p_2. Note that Fig. 3(c1) shows the EN paths starting from p_2 latches, but paths starting from p_1 latches can be analyzed similarly. Note also that there is no direct path from a p_3 latch to a CG cell because every p_3 latch is in the back-to-back latch group.

Our second modification (M2) is to selectively remove the latch in the CG cell that controls p_1 or p_3 latches, as illustrated in Fig. 3(c2). The latch in the CG cell for traditional FF-based designs is important because it prevents glitches on the clock signal. In 3-phase latch-based designs, on the other hand, the latch in the CG is redundant when its EN has no start point latched by the same phase as the CG cell. Consider a CG that controls a p_1 latch, if it has only paths starting from p_2 or p_3 latches, then upon the opening of the p_1 latch, all p_2 and p_3 latches have already closed. This means that EN will stabilize before p_1 latches open and, once stabilized, will stay stable until the next rising edge of p_3. In other words, EN is guaranteed to stay stable during p_1 is high and hazards are naturally avoided. In contrast, if a p_1 latch has a path to EN that controls a p_1 CG, then the latch in CG should not be removed to avoid potential hazards when p_1 latch is transparent.

Our second means of clock gating p_2 latches is data-driven clock gating (DDCG) [24]. In DDCG, the clock signal is gated when D and Q are the same and enabled only when D differs from Q. It saves dynamic power when the data pin has low switching activity [24]. In the structure of a single-bit DDCG, every individual latch requires an XOR and an AND gate. To reduce this overhead, multi-bit DDCG structure groups several latches to be driven by the same gated clock signal, generated by combining the comparison signals from individual latches. Since the clock signal p_2 drives one AND gate to control a group of latches, the clock tree can consume less power. However, the gating efficiency may be reduced because a switch in one latch enables the whole group. Therefore, it is beneficial to group latches whose switching activities are low and highly correlated. In this paper, after gating based on common upstream enables, we try to gate the remaining ungated p_2 latches using multi-bit DDCG. We first group latches according to their toggle rates and add DDCG to those groups whose data pins have low switching activities (less than 1% of the clock frequency). Note that the optimal fanout of a CG cell depends on the average toggling statistics of the individual registers as well as the process technology and cell library. In this work, we choose 32 as the maximum CG fanout and therefore groups with more latches are divided among multiple multi-bit DDCG structures.

Finally, we note that in some libraries, FFs or latches are physically merged into a single multi-bit cell so that the clock signals are shared among multiple storage elements. Coupling multi-bit registers with multi-bit clock gating may yield more power savings [25], but this is outside the scope of this paper.

V. Experimental Results

This section quantifies the benefits of the proposed conversion algorithm comparing the resulting 3-phase designs to the original FF-based as well as traditional master-slave latch-based designs. The experiments rely on an industrial 28-nm FDSOI CMOS cell library and a range of circuits that include, ISCAS89 benchmark circuits [16], CEP submodules [17], and three CPU designs, a 3-stage MIPS Open Core Plasma [18], a RISC-V Rocket Core [19], and an ARM-M0 core [20]. The ISCAS89 benchmarks were run at 1 GHz. Other circuits were run at 500 MHz, except for RISC-V and ARM-M0 which operate at 333.3 MHz. We validated both master-slave and 3-phase latch-based circuits by streaming inputs to the FF-based and latch-based designs and compare output streams.[1] These gate-level simulations were also used to determine signal activity that drove data-driven clock gating and to measure the relative power consumption of our approach. All experiments were run on two Intel Xeon E5-2450 v2 CPUs with 128GB of RAM.

Note that for a fair comparison, all variants of each design are run at the same frequency. With library-provided integrated clock gating cells, the tool decides the best clock gating technique for FF- and master-slave latch-based designs. The modified retiming strategy and relevant clock gating strategies described in Section IV are also performed on the latch-based designs.

Table I summarizes the number of registers (FFs/latches) and total area (μm^2) in the original FF-based, conventional master-slave latch-based, and 3-phase latch-based designs. The savings for the number of registers are the percentages of

[1]For ISCAS designs we used auto-generated pseudo-random input streams. For CEP and CPU designs, we used the open-source provided testbenches. In particular, Plasma was simulated using the “pi" program, ARM-M0 was simulated using “hello world", RISC-V was simulated using the “rv32ui-v-simple", and CEP designs were simulated using the open-source provided self-check programs.

TABLE I: Number of registers (FFs or latches) and Total Area (μm^2) in the original flip-flop (FF), converted master-slave latch (M-S), and proposed 3-phase (3-P) latch based designs

	Design	# of Regs					Total Area				
		FF	M-S	3-P	Save (%) 2*FF	Save (%) M-S	FF	M-S	3-P	Save (%) FF	Save (%) M-S
ISCAS	s1196	18	36	26	27.8	27.8	240	228	219	9.0	4.2
	s1238	18	36	26	27.8	27.8	238	229	215	9.7	6.1
	s1423	81	158	146	9.9	7.6	591	466	524	11.5	-12.4
	s1488	6	16	12	0.0	25.0	217	232	239	-10.2	-3.1
	s5378	163	317	250	23.3	21.1	1164	930	914	21.4	1.7
	s9234	140	278	225	19.6	19.1	902	752	741	17.8	1.5
	s13207	457	890	725	20.7	18.5	2675	2058	2056	23.1	0.1
	s15850	454	904	747	17.7	17.4	2885	2565	2315	19.7	9.7
	s35932	1728	3456	2737	20.8	20.8	11770	9356	9054	23.1	3.2
	s38417	1489	2751	2366	20.6	14.0	9395	7272	7863	16.3	-8.1
	s38584	1319	2633	2422	8.2	8.0	9355	7683	7961	14.9	-3.6
	Average	534	1043	880	17.8	18.8	3585	2888	2918	14.2	-0.1
CEP	AES	9715	16829	12871	33.8	23.5	133115	121960	119174	10.5	2.3
	DES3	436	842	573	34.3	31.9	2711	2738	2449	9.7	10.6
	SHA256	1574	3308	2523	19.9	23.7	9996	9461	8594	14.0	9.2
	MD5	804	1889	996	38.1	47.3	7023	6630	6947	1.1	-4.8
	Average	3132	5717	4241	31.5	31.6	38212	35197	34291	8.8	2.0
CPU	Plasma	1606	2357	2078	35.3	11.8	8944	7546	8029	10.2	-6.4
	RISCV	2795	5312	4084	26.9	23.1	14453	15268	14002	3.1	8.3
	ArmM0	1397	2713	2290	18.0	15.6	10690	11007	11514	-7.7	-4.6
	Average	1933	3461	2817	26.8	16.8	11362	11274	11182	1.9	-0.9
Average		1344	2485	1950	22.4	21.3	12576	11466	11267	11.0	0.8

the number of latches in 3-phase designs compared to twice the number of FFs in FF-based designs and compared to the number of latches in master-slave designs. The results show an average of 22.4% and 21.3% reductions, respectively. Notice that the 3-phase algorithm has the least benefit on ISCAS89 circuits and, in particular, no benefit on s1488, compared to FF-based designs. According to [26], s1488 is re-synthesized from a controller and may suggest that our algorithm brings limited benefits to control dominated designs that have a predominance of FFs with combinational feedback. In terms of the total area, 3-phase designs show average of 11% and 0.8% improvements compared to FF and master-slave designs, where the overall average is across individual benchmarks without biasing by size. The CPU benchmarks show a relatively low area saving compared to FF-based designs. This is the result of the fact that converting FF-to latch-based designs sometimes increases the combinational logic area depending on the results of retiming. Master-slave designs have more slave latches that can be moved around thus possibly better retiming results and smaller logic area, compared to 3-phase designs. The degree of logic area increase is clock frequency dependent and re-running these experiments at lower frequencies, reduces this impact.

Table II reports the power dissipation based on simulation. The total power is composed of three power groups, namely the clock network (Clock), register/sequential logic (Seq), and combinational logic (Comb). The 3-phase designs achieve an overall average of 15.5% and 18.5% power reductions compared to FF and master-slave designs. Note the average saving is calculated by taking the average across the benchmarks without weighting by their size. The combinational power savings drop from 15% to -4% in the comparison changing from FF to master-slave designs. This can be explained by

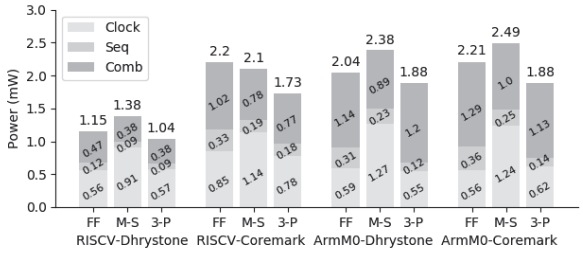

Fig. 4: Power dissipation (mW) of RISC-V and Arm-M0 running Dhrystone and Coremark with flip-flop, converted master-slave latch, and proposed 3-phase latch based designs

the fact that latch-based designs allow time-borrowing which relaxes the normal FF edge-to-edge timing requirements and often have less glitching and fewer hold buffers than their FF-based counterparts.

Coremark and Dhrystone are two standard benchmarks traditionally used to measure general process (CPU) performance. For this reason, we also tested our generated place-and-routed CPUs (RISC-V and Arm-M0) on both these benchmarks. Fig. 4 identifies the decomposition of power consumption in the middle and the total power consumption on the top of each bar. The 3-phase algorithm saves power by an average of 15.6% and 21.2% in RISC-V and 8.3% and 20.1% in Arm-M0 compared to FF and master-slave designs, respectively. Note that the obtained power savings are similar to that obtained using our simple CPU testbench programs, i.e., 9.0% and 26.6% for RISC-V and 7.9% and 36.6% for Arm-M0 compared to FF and master-slave designs.

The design flow for 3-phase designs, including ILP, conversion, retiming, and place-and-route, requires average of 204% and 44% more run-times compared to FF and master-slave designs across all designs, with most of the increase run-time occurring in place-and-route. In particular, the ILP solver consumes at most 27 seconds among all the benchmarks and is generally a tiny fraction (< 1%) of the overall run-time. In contrast, with three clocks to be routed, 3-phase designs take generally three times longer in clock tree synthesis and 35% longer in routing compared to FF-based designs. Nevertheless, these run-time increases are quite manageable, suggesting that our proposed approach is computationally practical for at least moderately-sized blocks.

In summary, our experiments suggest that while significant saving in area and power is possible with our proposed approach, the amount of savings is variable and likely depends on a combination of factors including 1) the percentage of FFs with combinational feedback that limits the savings in number of latches and 2) the impact in retiming latch-based designs on the combinational logic.

VI. Conclusions

This paper presents an algorithm to automatically convert a FF-based design into a 3-phase latch-based design that uses an ILP to minimize the number of required latches. The post PnR results on a broad range of benchmark circuits show significant

TABLE II: Power dissipation (mW) based on simulation of flip-flop (FF), master-slave latch (M-S), and 3-phase latch-based designs. We run ISCAS designs on pseudo-random input streams, CEP designs on open-source provided self-check programs, Plasma on "pi", RISC-V on "rv32ui-v-simple", and ARM-M0 on "hello world" program.

	Design	FF Power				M-S Power				3-Phase Power				Save (%) to FF				Save (%) to M-S			
		Clock	Seq	Comb	Total	Clock	Seq	Comb	Total	Clock	Seq	Comb	Total	Clock	Seq	Comb	Total	Clock	Seq	Comb	Total
ISCAS	s1196	0.08	0.04	0.18	0.30	0.09	0.04	0.18	0.32	0.07	0.03	0.18	0.28	12.29	22.28	1.68	7.12	24.92	24.84	0.87	11.06
	s1238	0.08	0.04	0.17	0.29	0.10	0.04	0.18	0.32	0.07	0.03	0.17	0.27	11.69	22.72	0.35	6.48	25.65	21.59	6.70	14.19
	s1423	0.56	0.08	0.17	0.82	0.42	0.08	0.12	0.63	0.50	0.11	0.15	0.75	11.04	-25.12	15.26	8.21	-17.40	-27.74	-21.96	-19.62
	s1488	0.03	0.01	0.13	0.17	0.04	0.02	0.13	0.19	0.03	0.01	0.12	0.17	-11.86	1.56	2.19	-0.06	27.27	22.99	3.63	10.61
	s5378	0.82	0.25	0.37	1.44	0.84	0.25	0.24	1.34	0.59	0.28	0.26	1.13	28.53	-15.32	31.16	21.75	30.33	-13.71	-5.28	15.61
	s9234	0.69	0.10	0.10	0.89	0.62	0.11	0.05	0.78	0.55	0.10	0.08	0.73	20.12	-4.18	22.80	17.72	11.58	4.03	-44.67	6.73
	s13207	2.04	0.43	0.42	2.89	1.98	0.50	0.20	2.69	1.53	0.46	0.22	2.21	25.10	-5.06	46.74	23.67	22.91	8.61	-8.27	17.87
	s15850	2.13	0.31	0.53	2.98	2.14	0.30	0.44	2.87	1.81	0.30	0.35	2.47	14.88	3.77	33.53	17.10	15.12	-0.70	19.04	14.10
	s35932	11.50	2.70	4.32	18.50	10.60	3.01	3.11	16.80	8.12	2.83	3.06	14.00	29.41	-4.59	29.21	24.32	23.42	6.20	1.48	16.67
	s38417	6.34	0.88	2.05	9.26	6.27	0.96	1.40	8.62	4.81	0.96	1.47	7.24	24.08	-9.58	28.36	21.83	23.25	-0.82	-4.87	16.03
	s38584	7.11	2.50	4.88	14.50	7.04	2.68	3.54	13.30	7.31	3.02	3.40	13.70	-2.84	-21.07	30.29	5.52	-3.83	-12.88	3.98	-3.01
	Average	2.85	0.67	1.21	4.73	2.74	0.73	0.87	4.35	2.31	0.74	0.86	3.90	14.77	-3.14	21.96	13.97	16.66	2.95	-4.49	9.11
CEP	AES	18.80	0.05	0.20	19.10	14.30	0.06	0.17	14.50	7.94	0.06	0.26	8.27	57.76	-20.50	-32.54	56.72	44.46	-10.31	-54.59	42.99
	DES3	0.26	0.14	0.51	0.91	0.21	0.12	0.41	0.74	0.20	0.10	0.41	0.72	21.75	25.98	19.98	21.42	5.13	9.98	0.27	3.18
	SHA256	0.13	0.05	0.13	0.31	0.27	0.06	0.09	0.42	0.13	0.05	0.13	0.30	-5.69	-0.22	7.26	0.82	50.13	17.69	-32.07	27.21
	MD5	0.11	0.02	0.28	0.40	0.38	0.19	1.21	1.78	0.09	0.02	0.25	0.36	18.58	-10.28	8.29	9.96	76.97	87.25	79.04	79.48
	Average	4.82	0.06	0.28	5.18	3.79	0.10	0.47	4.36	2.09	0.06	0.26	2.41	23.10	-1.25	0.75	22.23	44.17	26.15	-1.84	38.22
CPU	Plasma	0.59	0.44	0.65	1.68	0.99	0.19	0.45	1.63	0.64	0.17	0.54	1.36	-9.31	61.23	16.30	19.03	34.97	8.61	-20.73	16.54
	RISC-V	0.52	0.11	0.37	1.01	0.87	0.07	0.30	1.25	0.54	0.07	0.30	0.92	-4.15	33.19	20.26	8.99	37.70	2.71	0.30	26.63
	ARM-M0	0.54	0.31	1.14	2.00	1.23	0.23	1.34	2.90	0.50	0.11	1.22	1.84	6.74	63.50	-6.73	7.92	59.14	49.45	8.95	36.56
	Average	0.55	0.29	0.72	1.56	1.03	0.16	0.70	1.92	0.56	0.12	0.69	1.37	-2.24	52.64	9.94	11.98	43.93	20.26	-3.83	26.58
Average		2.91	0.47	0.92	4.30	2.69	0.49	0.75	3.95	1.97	0.49	0.70	3.15	13.78	6.57	15.24	15.47	27.32	10.99	-3.79	18.49

savings are possible in both area and power, particularly for pipelined circuits such as multi-stage CPUs when compared to both FF and master-slave latch-based designs.

Our future work includes quantifying these benefits associated with higher tolerance to PVT variations and increased robustness to hold failures. In addition, we plan to quantify the advantage of this approach when applied to soft-error and timing resilient templates in which the decrease in latches also reduces the overhead of the necessary error detection logic.

ACKNOWLEDGMENT

P. A. Beerel also consults for Galois, Inc. in the area of asynchronous design. This work was partially supported by NSF Grant #1619415 and DARPA Contract #HR001119C0070.

REFERENCES

[1] R. A. Haring, R. Bellofatto, A. A. Bright, P. G. Crumley, M. B. Dombrowa, S. M. Douskey, M. R. Ellavsky, B. Gopalsamy, D. Hoenicke, T. A. Liebsch, J. A. Marcella, and M. Ohmacht, "Blue gene/L compute chip: Control, test, and bring-up infrastructure," *IBM Journal of Research and Development*, vol. 49, no. 2.3, pp. 289–301, March 2005.

[2] K. Singh, H. Jiao, J. Huisken, H. Fatemi, and J. P. De Gyvez, "Low power latch based design with smart retiming," in *ISQED*, 2018, pp. 329–334.

[3] M. Pons, T. Le, C. Arm, D. Séverac, J. Nagel, M. Morgan, and S. Emery, "Sub-threshold latch-based icyflex2 32-bit processor with wide supply range operation," in *ESSDERC*, Sept 2016, pp. 33–36.

[4] A. P Hurst and R. K Brayton, "The advantages of latch-based design under process variation," in *Proceedings of the IWLS*, 2006.

[5] M. Fojtik, D. Fick, Y. Kim, N. Pinckney, D. Harris, D. Blaauw, and D. Sylvester, "Bubble razor: An architecture-independent approach to timing-error detection and correction," in *ISSCC*, 2012, pp. 488–490.

[6] D. Hand, M. T. Moreira, H.-H. Huang, D. Chen, F. Butzke, Z. Li, M. Gibiluka, M. Breuer, N. L. V. Calazans, and P. A. Beerel, "Blade – a timing violation resilient asynchronous template," in *ASYNC*, 2015, pp. 21–28.

[7] J.-F. Lin, "Low-power pulse-triggered flip-flop design based on a signal feed-through scheme," *IEEE TVLSI*, vol. 22, no. 1, pp. 181–185, 2014.

[8] S. Paik, L.-e. Yu, and Y. Shin, "Statistical time borrowing for pulsed-latch circuit designs," in *ASP-DAC*, 2010, pp. 675–680.

[9] K. Singh, O. A. R. Rosas, H. Jiao, J. Huisken, and J. P. de Gyvez, "Multi-bit pulsed-latch based low power synchronous circuit design," in *ISCAS*, 2018, pp. 1–5.

[10] Y. Ding, W. Jin, G. He, and W. He, "Short path padding with multiple-Vtcells for wide-pulsed-latch based circuits at ultra-low voltage," in *ASICON*, Oct 2017, pp. 985–988.

[11] Y. Shin and S. Paik, "Pulsed-latch circuits: A new dimension in asic design," *IEEE D&T of Comp.*, vol. 28, no. 6, pp. 50–57, 2011.

[12] K. Yoshikawa, Y. Hagihara, K. Kanamaru, Y. Nakamura, S. Inui, and T. Yoshimura, "Timing optimization by replacing flip-flops to latches," in *ASP-DAC*, 2004, pp. 186–191.

[13] J. Cortadella, A. Kondratyev, L. Lavagno, and C. P. Sotiriou, "Desynchronization: Synthesis of asynchronous circuits from synchronous specifications," *IEEE TCAD*, vol. 25, no. 10, pp. 1904–1921, 2006.

[14] H. Cheng, H.-L. Wang, M. Zhang, D. Hand, and P. A. Beerel, "Automatic retiming of two-phase latch-based resilient circuits," *IEEE TCAD*, 2018.

[15] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun, "Optimal clocking of synchronous systems," in *ACM Intl. Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, 1990, pp. 1–21.

[16] "ISCAS89: International symposium on circuits and systems sequential benchmark. http://www.pld.ttu.ee/~maksim/benchmarks/iscas89/verilog/."

[17] "MIT-LL common evaluation platform (CEP)," https://github.com/mit-ll/CEP, available: 2019.

[18] "Plasma CPU," http://opencores.org/project,plasma, available: 2014.

[19] "Rocket chip," https://github.com/freechipsproject/rocket-chip, available: 2016.

[20] "ARM Cortex M0," https://developer.arm.com/products/processors/cortex-m/cortex-m0.

[21] H. Cheng, Y. Gu, and P. A. Beerel, "Automatic conversion from flip-flop to 3-phase latch-based designs," *CoRR*, vol. abs/1906.10666, 2019, presented at IWLS2019. [Online]. Available: http://arxiv.org/abs/1906.10666

[22] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2018. [Online]. Available: http://www.gurobi.com

[23] D. Chinnery and K. Keutzer, *Closing the gap between ASIC & custom: tools and techniques for high-performance ASIC design*. Springer Science & Business Media, 2002.

[24] A. G. Strollo, E. Napoli, and D. De Caro, "New clock-gating techniques for low-power flip-flops," in *ISLPED*, 2000, pp. 114–119.

[25] D. Gluzer and S. Wimer, "Probability-driven multibit flip-flop integration with clock gating," *IEEE TVLSI*, vol. 25, no. 3, pp. 1173–1177, 2016.

[26] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *ISCAS*, 1989, pp. 1929–1934.