2011 21st International Conference on Field Programmable Logic and Applications

Latch-Based Performance Optimization for FPGAs

Bill Teng
Department of ECE
University of Toronto
Toronto, ON, Canada
Email: bill.teng@eecg.toronto.edu

Jason H. Anderson
Department of ECE
University of Toronto
Toronto, ON, Canada
Email: janders@eecg.toronto.edu

***Abstract*—We explore using pulsed latches for timing optimization – a first in the FPGA community. Pulsed latches are transparent latches driven by a clock with a non-standard (non-50%) duty cycle. We exploit existing functionality within commercial FPGA chips to implement latch-based optimizations that do not have the power or area drawbacks associated with other timing optimization approaches, such as clock skew and retiming. We propose an algorithm that iteratively replaces certain flip-flops in a logic design with latches for an improvement in circuit speed. Results show that much of the performance improvement achieved by using *multiple* skewed clocks can also be achieved using a *single* clock and latches. We also consider the impact of short delay paths (i.e. minimum delays), which can cause hold-time violations. Under conservative minimum delay assumptions, our latch-based optimization, operating on the routed design, provides a 5% performance improvement, on average, essentially for "free" (i.e. without any re-routing/delay padding). We show that short paths greatly hinder the ability of using latches for speed improvement, motivating further work to reduce their effects.**

I. Introduction

The speed of a sequential circuit is inversely related to the length of the longest path between any two pairs of flip-flops. *Time borrowing* or cycle stealing is a well-studied technique that allows the longest paths to "steal" time from adjacent combinational stages, in order to reduce the clock period.

One well-known approach, clock skew scheduling, intentionally delays some clocks to certain flip-flops to steal time from subsequent combinational stages [1]. For example, for a combinational logic path from a flip-flop j to flip-flop i, clock skew scheduling may delay the arrival of the clock signal to i, thereby allowing more time for a logic signal to arrive at i's input. Recent research [2]–[4] has shown that only a few skewed clocks are necessary to obtain appreciable improvements in circuit speed. Unfortunately, clocks comprise 20-39% of dynamic power consumption in commercial FPGAs [5], [6] and FPGAs already consume 7-14× more dynamic power than ASICs [7]. It is clear that for FPGAs to remain competitive with ASICs, it would be desirable to improve circuit performance without using extra clocks.

Another approach to borrowing time is *retiming*, which physically relocates flip-flops across combinational logic to balance the delays between combinational stages. Although extra clock lines are not required to borrow time, the practical usage of retiming is limited due to its impact on the verification methodology, i.e., equivalence checking and functional simulation [2]. Retiming can change the number of flip-flops in a design, for example, in the case of moving a flip-flop "upstream" from the output of a multi-input logic gate to its inputs. As such, retiming may increase circuit area, and make it difficult for a designer to verify functionality or to correlate the retimed design with the original RTL specification.

In this work, we use latches to perform time borrowing, thereby overcoming the power barrier associated with using multiple clocks, and the netlist modifications required for retiming. Consider again a combinational path from a flip-flop j to a transparent latch i. The maximum allowable delay for the path is extended beyond the clock period. Specifically, a transition launched from j need not settle on i's input by the next rising clock edge – it may settle after the clock edge during the time window when i is transparent. A downside of latches, however, is that they make timing analysis more difficult because transparency allows critical long (max delay) and short (min delay) paths to extend across multiple combinational stages, unlike standard timing analysis using flip-flops. This can lead to hold-time violations. Using *pulsed latches* driven by a clock with a non-standard duty cycle or *pulse width* (i.e. not 50%), is one method of reducing the effects of short paths plaguing conventional latch-based circuits, while allowing time borrowing for long paths. This is a viable option as commercial FPGAs (e.g. Xilinx Virtex 6) can generate clocks with different duty cycles, as well as allow the sequential elements in Combinational Logic Blocks (CLBs) to be used as either flip-flops *or* latches [8], [9]. That is, commercial FPGAs *already* contain the necessary hardware functionality to support pulsed latch-based timing optimization, but to the best of our knowledge, no prior work has explored the pulsed latch concept for FPGAs.

An illustration of pulsed latches is shown in Fig. 1. Fig. 1 (a) shows that with latches, it is possible that two signals launched on two different clock cycles can arrive at one flip-flop (FF3) at the same time, which clearly is invalid. The solid and dashed lines represent long and short combinational paths, respectively, between latch L2, FF1, and FF3. The cause of this problem is the short path signal launched from FF1 arriving at L2 when it is still transparent – a hold-time violation. Fig. 1(b) shows that this violation can be fixed by reducing the pulse width until the short signal arrives at L2 when it is no longer transparent. Naturally, the use of large pulse widths is desirable in the sense that they permit more time borrowing between

978-0-7695-4529-5/11 $26.00 © 2011 Crown Copyright
DOI 10.1109/FPL.2011.21

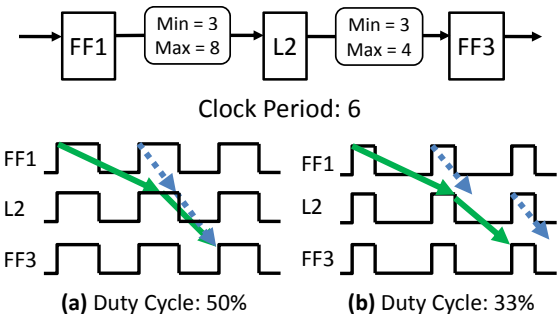

Fig. 1. Illustration showing how a 33% duty cycle can fix hold-time violation.

adjacent combinational stages; however, large pulse widths are more likely to create hold-time violations. In our approach, we reduce the negative impact of short paths by *forcing* some flip-flops to remain as flip-flops. The main contributions of our work are:

- An algorithm that improves circuit speed by selectively converting some flip-flops to latches in FPGAs.
- Results showing that when short path (min) delays are 70% of long path (max) delays, our optimization, operating on the routed design, improves circuit speed by 5%, on average – a "no cost" performance gain.
- We compare the use of pulsed latches (driven by a *single* clock) with clock skew using multiple clocks, and show that pulsed latches provide the majority of benefits offered by clock skew.

The remainder of this paper is organized as follows: In Section II, we summarize related work on clock skew and latches. In Section III, we review latch timing constraints. In Section IV, we show how the latch timing constraints can be represented as a graph and formulate the timing optimization problem in a graph-theoretic manner. We explain our algorithm in Section V. We present our results in Section VI. Finally, we conclude and provide insight into future directions of our work in Section VII.

II. Related Work

Although we could not find published work that uses latches, time borrowing using clock skew has been explored. Singh and Brown showed that 4 shifted clock lines provides over a 20% improvement in circuit speed [4]. The downside to this approach is the use of extra clock lines to allow time borrowing, which increases overall power consumption.

Other work [10], [11] involving FPGAs has focused on the use of programmable delay elements $(PDEs)$ to purposely delay clock signals. The work in [10] used PDEs on the clock tree, whereas the PDEs were inserted into FPGA logic elements in [11]. Both methods incur a hardware penalty and require additional architectural considerations.

Timing optimization of latch-based circuits has been studied extensively for ASICs. Most prior work has represented the latch-based optimization problem using linear constraints and solved it using linear programming (LP) [12]–[14], [25] or graph algorithms [15], [24]. Among the prior work on latches, our approach is most similar to [24]. The authors optimize circuit performance by using two clocks with adjustable duty cycles. Their approach is exact and can be extended to more than two clocks. However, they strictly forbid combinational paths that start and end at the same latch, which we found to be quite prevalent in our benchmark suite. Our formulation supports these combinational paths, while also improving performance with only a single clock.

Pulsed latches have recently been explored in the ASIC context by Lee et. al [16], [17]. Using flip-flop-like timing constraints, their optimization strategy relies on exploiting the difference between pulse widths and clock delays to steal time from neighboring combinational stages. Their approach to time borrowing uses multiple pulse widths and skewed clocks. This differs from our approach which mimics the presence of multiple "skews" using *one* pulse width.

T_{cq}	clock-to-Q delay
T_{dq}	data-to-Q delay
a_i, A_i	earliest and latest arrival times at latch i
P	clock period
cd_{ji}, CD_{ji}	short and long $j \rightarrow i$ combinational path delay from latch j to latch i
T_{su}	setup time
T_h	hold-time
W_i	pulse width of latch i

TABLE I
Summary of Latch Timing Parameters.

III. Latch Timing Constraints

Table I summarizes variables necessary to understand latch timing constraints. The main difference between latch and flip-flop timing constraints is the ability of short and long paths to extend through multiple latches. This manifests as non-linearity (a *max* function) in the constraints [17]:

$$A_i = max_{\forall j \rightarrow i}[max(T_{cq}, A_j + T_{dq}) + CD_{ji}], \forall i \quad (1)$$

$$a_i = min_{\forall j \rightarrow i}[max(T_{cq}, a_j + T_{dq}) + cd_{ji}], \forall i \quad (2)$$

Equations (1) and (2) show that the latest and earliest arrival times of latch i are functions of the latest and earliest arrival times of latches connected to i through combinational paths, respectively. The latch setup constraint shown below assumes that latch i is driven by a pulse of width W_i [17]:

$$A_i \leq P + W_i - T_{su}, \quad (3)$$

Combining (1) and (3), we obtain:

$$max_{\forall j \rightarrow i}[max(T_{cq}, A_j + T_{dq}) + CD_{ji}] \leq P + W_i - T_{su}, \forall i \quad (4)$$

We can remove the leftmost max term in (4) by generating (4) for every $j \rightarrow i$ path:

$$max(T_{cq}, A_j + T_{dq}) + CD_{ji} \leq P + W_i - T_{su}, \forall j \rightarrow i \quad (5)$$

The purpose of the remaining max term is to ensure that the signal at latch j launches no earlier than T_{cq} after the rising edge. We can represent (5) with two linear constraints:

$$A_j + T_{dq} + CD_{ji} \leq P + W_i - T_{su}, \forall j \rightarrow i \quad (6)$$

$$A_j + T_{dq} \geq T_{cq}, \forall j \quad (7)$$

(7) is a lower bound on the launch time of a signal from latch j. Conservatively, we can assume that the latest arrival time of latch j always occurs at the falling edge of a pulse, that is $A_j = W_j - T_{su}$. Plugging this into (6) and (7) gives:

$$W_j + T_{dq} + CD_{ji} \leq P + W_i, \forall j \rightarrow i \quad (8)$$

$$W_j - T_{su} + T_{dq} \geq T_{cq}, \forall j \quad (9)$$

Similarly, the hold-time constraint for latches, when combined with (2) yields:

$$min_{\forall j \rightarrow i}[max(T_{cq}, a_j + T_{dq}) + cd_{ji}] \geq W_i + T_h, \forall i \quad (10)$$

We can relax this non-linear constraint by first transforming (10) to occur between every latch pair connected by a combinational path. We conservatively assume that every early signal launches at the beginning of a latch's opening window (i.e. the rising edge of the clock). If we assume $T_{cq} \geq T_{dq}$ and set $a_j = 0$, this results in:

$$T_{cq} + cd_{ji} \geq W_i + T_h, \forall j \rightarrow i \quad (11)$$

Although the assumptions made to remove the non-linearity from (8) and (11) would appear to restrict the full potential of using latches, we will show that one clock can implement most of the gains achieved by clock skew requiring multiple clock lines.

IV. Graph formulation

We can formulate the problem of latch-based timing optimization as a graph problem. This section will discuss how constraints derived in Section III map into a graph problem. A proof of this equivalence can be found in [18]. Let $G = (V, E)$ be a strongly-connected directed graph. Let a *vertex* $v \in V$ represent a flip-flop or a latch in G. Every v has an associated W_v, the pulse width. Let an *edge*, $e(u, v)$, and its delay, $d\,(u, v)$ represent the delay on a $u \rightarrow v$ combinational path.

A *path* is a traversal of vertices through connecting edges with an arbitrary start and end vertex. A *cycle* is a path that starts and ends at the same vertex. Let *c* and *C* represent a cycle and the set of *all* cycles in G, respectively. The *maximum cycle ratio* (MCR) is defined as:

$$MCR\,(G) = max_{c \in C} \left[\sum_{e(u,v) \in c} \frac{d\,(u, v)}{|c|} \right] \quad (12)$$

where $|c|$ represents the number of edges on c. The MCR is the optimum clock period, P, for the circuit represented by G. In essence, the MCR is the maximum total delay of any cycle in G divided by the number of sequential elements on that cycle. It represents the best clock period that can be achieved by latches/retiming/clock skew for the circuit, given the specified delay values. A proof of this property can be found in [19]. Given P, the set of pulse widths for every latch, $\omega = \{W_0, W_1, \ldots, W_{|V|-1}\}$, can be determined (outlined below).

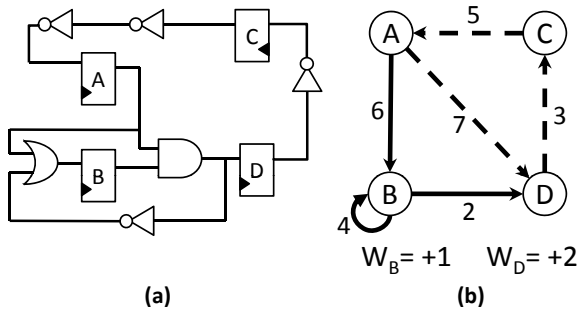

Fig. 2. Sample circuit fragment and its graph representation. The dashed cycle in (b) yields the MCR.

Fig. 2 shows how a circuit fragment, (a), is represented in our graph formulation, (b), where the integers next to edges represent path delays in the circuit in (a). Fig. 2 (b) also illustrates how the $MCR\,(G)$ yields the optimum P. For this example, assume that every edge in (b) represents constraint (8). The cycle containing dashed edges, $v_A \rightarrow v_D \rightarrow v_C \rightarrow v_A$, is the MCR in this example, with a value of 5: the sum of edge delays along the cycle is 15, and 15/(3 edges) = 5. Thus, we can operate this circuit with a clock period of 5. Observe, however, that the edge from $v_A \rightarrow v_D$ has a path delay of 7. Consequently, we must set $W_D = +2$, in order for signals on the $v_A \rightarrow v_D$ path to arrive on time. In addition, since timing analysis requires *EVERY* constraint to be satisifed, we must set $W_B = +1$ to satisfy $v_A \rightarrow v_B$. Any other cycle in this graph, with a lower cycle ratio would not satisfy every constraint. Note that in this example, constraints (9) and (11) have been omitted for clarity. However, they can also be represented in the graph formulation.

The equations and constraints defined in Section III can be solved using linear programming. However, we use Howard's algorithm [20] to find the MCR. Howard's algorithm maintains a subgraph, $G_{sub} \subseteq G$, where each vertex has exactly one outgoing edge. Since G is strongly-connected, G_{sub} is also. This implies G_{sub} contains at least one cycle. A cycle is found in G_{sub} by traversing its edges and its cycle ratio, r_{sub}, is used as an initial estimate for MCR(G). To determine if r_{sub} is the MCR, Bellman-Ford is applied to G. If r_{sub} is not the MCR, G_{sub} is "improved upon" and the algorithm continues until the MCR is found. The interested reader is referred to [21] for details. Although the algorithm has no known upper bound on the worst-case time complexity, it has an almost linear execution time in practice.

V. Algorithm

Although we can incorporate (11) into G (i.e. hold-time constraints) and find the optimum P using the MCR approach, the solution to this problem would imply that every sequential element is a latch with a (possibly different) pulse width,

Input: $G(V, E)$, E_{min}
Output: $P_{final}, W_{final}, \omega_{final}$
1: $P_{init}, \omega_{init} \leftarrow Howard\,(G)$
2: Sort edges in E in ascending order of their short path delays in E_{min}
3: $W_{final} \leftarrow max(\omega_{init})$
4: **for** $e(u, v) \in$ sorted E **do**
5: $d_{min}(u, v) \leftarrow$ short path delay of $e(u, v)$ from E_{min}
6: **if** $T_{cq} + d_{min}(u, v) < \omega_{init}(v) + T_h$ **then**
7: $W_{final} \leftarrow T_{cq} + d_{min}(u, v) - T_h$
8: **EXIT FOR**
9: **end if**
10: **end for**
11: **for** $e(u, v) \in E$ **do**
12: $d_{min}(u, v) \leftarrow$ short path delay of $e(u, v)$ from E_{min}
13: **if** $T_{cq} + d_{min}(u, v) < W_{final} + T_h$ **then**
14: $forceFlipFlop(v)$
15: **else**
16: $forceLatch(v)$
17: **end if**
18: **end for**
19: $P_{final}, \omega_{final} \leftarrow Howard\,(G)$

Fig. 3. Pulsed latch timing optimization.

thereby requiring multiple clocks and significant power consumption. We wish to use a single clock and therefore, must decide the specific pulse width to use, and also whether each sequential element should be a flip-flop or a latch. Although flip-flops cannot borrow time, their hold-time constraints are less restrictive and because of this, we use flip-flops to prevent very short paths from limiting the pulse width. The flip-flop/latch choice adds a binary decision element to the optimization problem, and would require Mixed Integer Linear Programming (MILP) to solve exactly. As MILP is NP-hard [18], unlike linear programming and Howard's algorithm, our algorithm is a greedy heuristic that attempts to maximize the pulse width in an iterative manner.

We first solve for the best-case clock period, P_{init}, and associated pulse widths, ω_{init}, without considering hold-time constraint (11), and then use ω_{init} in conjunction with (11) to guide the process of selecting some sequential elements to be latches, some to remain as flip-flops and settle on a *single* pulse width W_{final}. The approach we take is to start with a large value for W_{final} and then scale it back based on any short path delay violations. We then assign each sequential element to be either a flip-flop or a latch, and then re-solve for P and W_{final} based on the flip-flop/latch assignments. The full algorithm is detailed in Fig. 3.

The inputs to the algorithm are $G(V, E)$, which represents the constraint set depicted in Fig. 2, and the set of minimum delays between every pair of sequential elements, E_{min}. At line 1, we begin by calculating P_{init} and ω_{init} subject to only the constraints described in Section IV (no consideration of short path delays). At line 2, we sort the edges in E in ascending order of the values in E_{min}. This step identifies the first possible hold-time violation without having to iterate through all of E_{min}. Line 3 initializes W_{final} to be the maximum pulse width in the set ω_{init}. Lines 4 - 10 look for the first violation of constraint (11). If we do find a violation, we know that using a pulse width larger than $T_{cq} + d(u, v) - T_h$ would not satisfy the current violated constraint. Furthermore, subsequent edges in the sorted edge set would not generate a more constraining pulse width because the edge set is sorted in ascending order. After setting W_{final} at line 7, iteration through E stops at line 8. Lines 11 - 18 constrain sequential elements that have incoming edges with minimum delay less than W_{final} to be flip-flops to prevent other hold-time violations from occurring. We allow sequential elements that do not have incoming edges with delay less than W_{final} to be latches.

Finally, we compute P_{final} and ω_{final} subject to the newly-created constraints at line 19. In the next section, we show this approach delivers most of the performance benefits of clock skew using only a *single* clock with pulse width W_{final}.

VI. Experimental Study

We implemented our approach within the VPR 5.0 framework [22]. Our results use a mix of VPR 5.0 and MCNC benchmarks mapped to an architecture using a LUT size of 6, and cluster size of 8. We used an island style FPGA architecture consisting of 50% length-4 and 50% length-2 routing segments. Each benchmark was given an extra 30% in channel capacity on top of the minimum required for routing, in order to emulate a "medium stress" scenario. The results for each circuit are averaged over 5 runs, each using a different placement seed value. We set $T_h = \frac{1}{2}T_{cq}$, and $T_{cq} = T_{dq} = T_{su}$, based on values for the Xilinx Virtex 6.

In this work, we show the impact of latches and clock skew both with and without considering hold-time violations. Naturally, smaller short path delays lead to more hold-time violations and degraded performance results. However, the VPR timing model does not incorporate short path delays for combinational logic and routing paths. To handle this, we present results for several short path delay scenarios, ranging from optimistic to pessimistic. Specifically, we use VPR's timing information to calculate the shortest paths between sequential elements and emulate different scenarios by taking a fraction, *f*%, of these short delays. We consider three settings for f: 80% (optimistic), 70% (medium), and 60% (pessimistic).

Table II contrasts the gains of using pulsed latches and flip-flops (columns labeled PL), clock skew using flip-flops only (columns labeled CS) with no restrictions on the number of clock lines available, and theoretical possible gains using latches (column labeled PL_{opt}) without *any* short path constraints. The column labeled "Critical Path" presents the clock period of each circuit assuming only flip-flops are used, and the flops are driven by a single clock. We refer to this as the "traditional" flow. The results in the table, in essence, represent different ways of analyzing a set of fully placed and routed designs. Values in the table represent achievable clock periods in ns, as reported by VPR and our latch-based timing analysis

Minimum Delay Assumptions	None		80%		70%		60%	
Clock Period (ns)	Critical Path	PL_{opt}	PL	CS	PL	CS	PL	CS
bigkey	2.806	2.296	2.331	2.531	2.361	2.561	2.405	2.605
cf_cordic_v_18_18_18	5.177	2.952	4.834	4.834	4.889	4.889	4.944	4.944
cf_cordic_v_8_8_8	3.011	1.807	2.651	2.668	2.706	2.723	2.762	2.778
cf_fir_24_16_16	12.147	4.584	11.804	9.706	11.859	9.733	11.915	9.761
cf_fir_3_8_8	10.717	4.277	10.373	7.058	10.429	7.1	10.484	7.141
clma	7.434	6.954	7.113	7.154	7.095	7.154	7.129	7.156
des_area	5.644	5.342	5.342	5.544	5.342	5.544	5.342	5.544
des_perf	3.318	2.312	2.893	3.02	2.939	3.064	2.989	3.108
diffeq	4.714	3.287	4.37	4.37	4.426	4.426	4.481	4.481
diffeq_paj_convert	37.381	37.039	37.067	37.239	37.105	37.239	37.15	37.239
elliptic	4.421	3.433	4.077	4.077	4.133	4.133	4.188	4.188
fir_scu_rtl_restructured_for_cmm_exp	14.8	5.182	14.457	14.457	14.512	14.512	14.568	14.568
iir	16.026	6.064	15.682	15.311	15.738	15.413	15.793	15.514
iir1	15.384	14.16	15.04	15.04	15.096	15.096	15.151	15.151
mac1	12.258	11.503	11.914	11.914	11.97	11.97	12.025	12.025
mac2	21.012	20.275	20.662	20.668	20.717	20.724	20.773	20.779
oc54_cpu	18.073	11.552	17.73	17.73	17.785	17.785	17.84	17.84
paj_boundtop_hierarchy_no_mem	5.867	5.207	5.524	5.535	5.579	5.581	5.635	5.635
paj_framebuftop_hierarchy_no_mem	4.416	3.089	4.073	4.073	4.128	4.128	4.184	4.184
paj_raygentop_hierarchy_no_mem	11.407	5.564	11.063	7.639	11.119	7.683	11.174	7.726
rs_decoder_1	14.071	13.246	13.728	13.728	13.783	13.783	13.839	13.839
rs_decoder_2	18	17.409	17.657	17.669	17.713	17.714	17.768	17.768
s38417	4.918	4.225	4.574	4.574	4.629	4.629	4.685	4.685
s38584.1	4.542	4.424	4.424	4.424	4.424	4.424	4.424	4.424
sv_chip0_hierarchy_no_mem	4.193	3.419	3.85	3.845	3.905	3.9	3.96	3.956
sv_chip1_hierarchy_no_mem	12.277	3.518	11.933	8.832	11.989	8.86	12.044	8.888
tseng	4.752	3.198	4.409	4.409	4.464	4.464	4.52	4.52
Geomean	8.172	5.593	7.736	7.428	7.792	7.482	7.853	7.537
Ratio to Critical Path	1	0.684	0.947	0.909	0.954	0.916	0.961	0.922
Ratio to PL_{opt}		1	1.383	1.328	1.393	1.338	1.404	1.348
Ratio of Clock Skew to Pulsed Latches				0.96		0.96		0.96

TABLE II

ACHIEVABLE CLOCK PERIOD (NS) USING FLIP-FLOPS WITHOUT ANY TIME BORROWING (CRITICAL PATH), THEORETICAL MINIMUM WITH TIME BORROWING USING PULSED LATCHES(PL_{opt}), PULSED LATCHES (PL) AND CLOCK SKEW (CS) SUBJECT TO DIFFERENT MINIMUM DELAY ASSUMPTIONS.

framework. Geometric mean results and normalized geometric means appear at the bottom of each column.

On average, the "PL_{opt}" column shows that clock periods are reduced by 32%. Since this result does not consider hold-time violations, it represents a lower bound on the achievable clock period.

The rest of the results in Table II consider hold-time violations. For example, the columns grouped under 80% represent results for the case of minimum path delays set to be 80% of maximum path delays. Observe that the performance results degrade considerably when hold-times constraints must be honored, and underscores the necessity of considering such constraints. Pulsed latches provide 5% performance improvement, on average, relative to the traditional flow; clock skew with arbitrarily many clocks provides 9% improvement. Similar results are observed when minimum delays are set to 70% of maximum delays. Essentially, with 70% min delays, we get a 5% clock period improvement, on average, by latch conversion on a routed design, without requiring *any* delay padding (hold-time violation repair).

Comparing the results for pulsed latches and clock skew with 70% min delays, we observe identical clock periods for many circuits. A detailed analysis revealed that the circuits for which pulsed latches did poorly compared to clock skew contained what we term as *self-loops*: combinational paths that begin and end at the same sequential element. Such loops do not present a problem for clock skew with edge-triggered flip-flops. However, they do present a problem for latches, and limit the performance improvement for some benchmarks.

After observing that short paths have a considerable negative impact on *both* pulsed latch and clock skew-based optimization, we investigated the extent of delay padding needed to improve the clock period results. Specifically, we investigated how each circuit's clock period would be affected if it were possible to pad x% of its paths[1]. The aggregated results, normalized to the clock period without any short path repairs, are given in Fig. 4. The rightmost bar indicates that about 3% additional reduction in clock period would be achieved if it were possible to lengthen about 10% of paths. Circuit-by-circuit results could not be included due to page limits, however, we did see reductions of up to 8% for some circuits with 10% of paths delay padded.

The results in Table II are based on placement/routing solutions generated in a manner that was unaware of pulsed latch-based optimization. We wondered whether improved results could be achieved if the placement and routing tools were aware of latches? VPR gauges the criticality of a connection using its *slack ratio*, which is the ratio of the worst-case path delay through a connection, to the current critical path delay. We altered VPR's criticality notion to drive place and route using *cycle-slacks* [19], [23]. The cycle-slack ratio of a connection is the cycle ratio of any cycle that uses the connection, to the current MCR of the design. Its definition is thus analogous to slack ratio, making it straightforward to integrate into VPR. Fig. 5 shows the benefits of pulsed latch optimization with place and route using cycle-slacks, and our

[1]For this analysis, we removed circuits that already achieved the optimum clock period using latch-based optimization.

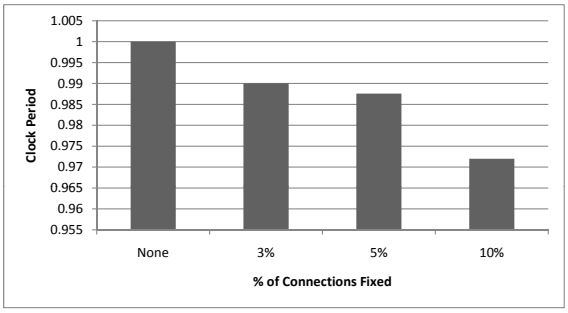

Fig. 4. Clock period as a result of extending 3%, 5%, or 10% of the total number of short paths under 70% minimum delay assumptions.

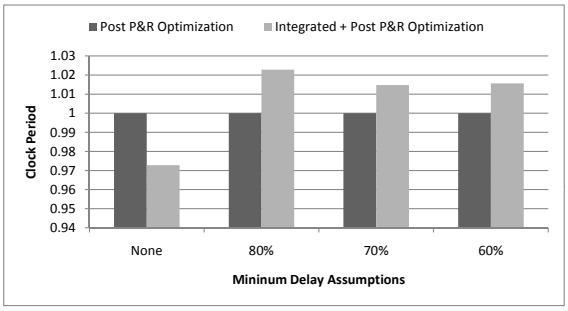

Fig. 5. Comparing the benefits of pulsed latches with and without using cycle-slacks in VPR.

optimization with regular place and route, normalized to 1.

If we ignore short paths completely, we observe a 3% reduction in the MCR when VPR is driven with cycle-slacks. However, with minimum delay constraints, our results indicate an increase in the MCR – i.e. slightly worse results than if VPR is driven by normal slacks! We believe this is a consequence of cycle-slacks being unaware of short paths, which constrain the amount of time borrowing allowed. While cycle-slacks indeed yield higher performance when short paths are not considered, higher performance requires more time borrowing to be realized, and is therefore more susceptible to short path violations. We noticed that VPR had to deal with many near critical cycles throughout the flow, and since it is difficult to equally optimize all equally-critical cycles, cycle-slacks may drive VPR to optimize the wrong cycles. Altering VPR to target critical cycles requiring a lot of time borrowing and lengthening paths in certain cases are likely fruitful directions for improving the results.

VII. Conclusions and Future Work

In this paper, we considered performance optimization using pulsed latches for FPGAs. We showed that most of the gains of using skewed clocks can be realized by a post place and route latch-based optimization that has no area or power drawbacks by using existing features found on commercial FPGAs. However, short paths still limit the possible gains of pulsed latches.

Results in Fig. 5 and Fig. 4 show that padding short paths can give considerable benefits, while driving VPR with cycle-slacks gives varied results. It is clear that optimizing the MCR throughout place and route is a flawed approach if short paths are not considered. This calls for an integrated approach that exploits the time borrowing benefits of pulsed latches and mitigates the short paths that hinder the benefits of pulsed latches.

References

[1] J. Fishburn, "Clock skew optimization," *Computers, IEEE Transactions on*, vol. 39, no. 7, Jul. 1990.
[2] K. Ravindran, A. Kuehlmann, and E. Sentovich, "Multi-domain clock skew scheduling," in *ACM/IEEE ICCAD*.
[3] J. Casanova and J. Cortadella, "Multi-level clustering for clock skew optimization," in *ACM/IEEE ICCAD*, 2009.
[4] D. P. Singh and S. D. Brown, "Constrained clock shifting for field programmable gate arrays," in *ACM FPGA*, 2002.
[5] L. Shang, A. S. Kaviani, and K. Bathala, "Dynamic power consumption in Virtex-II FPGA family," in *ACM FPGA*, 2002.
[6] V. Degalahal and T. Tuan, "Methodology for high level estimation of FPGA power consumption," in *IEEE/ACM ASP-DAC*, vol. 1, 2005.
[7] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," in *ACM FPGA*, 2006.
[8] *Virtex-6 FPGA Configurable Logic Block*, Xilinx, Inc., San Jose, CA, 2009.
[9] *Virtex-6 FPGA Clocking Resources*, Xilinx, Inc., San Jose, CA, 2011.
[10] C.-Y. Yeh and M. Marek-Sadowska, "Skew-programmable clock design for FPGA and skew-aware placement," in *ACM FPGA*, 2005.
[11] X. Dong and G. Lemieux, "PGR: Period and glitch reduction via clock skew scheduling, delay padding and glitchless," in *IEEE FPT*, 2009.
[12] K. A. Sakallah, T. N. Mudge, and O. A. Olukotun, "Analysis and design of latch-controlled synchronous digital circuits," in *ACM/IEEE DAC*, 1990.
[13] T. G. Szymanski, "Computing optimal clock schedules," in *ACM/IEEE DAC*, ser. DAC '92.
[14] B. Taskin and I. Kourtev, "Delay insertion method in clock skew scheduling," *IEEE TCAD*, vol. 25, no. 4, 2006.
[15] N. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Graph algorithms for clock schedule optimization," in *ACM/IEEE ICCAD*, 1992.
[16] S. Lee, S. Paik, and Y. Shin, "Retiming and time borrowing: Optimizing high-performance pulsed-latch-based circuits," in *ACM/IEEE ICCAD*, 2009.
[17] H. Lee, S. Paik, and Y. Shin, "Pulse width allocation and clock skew scheduling: Optimizing sequential circuits based on pulsed latches," *IEEE TCAD*, vol. 29, no. 3, 2010.
[18] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 2nd ed. McGraw-Hill Higher Education, 2001.
[19] "Maximum mean weight cycle in a digraph and minimizing cycle time of a logic chip," *Discrete Applied Mathematics*, vol. 123, no. 1-3.
[20] J. Cochet-terrasson, G. Cohen, S. Gaubert, M. M. Gettrick, and J. pierre Quadrat, "Numerical computation of spectral elements in max-plus algebra," 1998.
[21] A. Dasdan, "Experimental analysis of the fastest optimum cycle ratio and mean algorithms," *ACM TODAES*, vol. 9, no. 4, 2004.
[22] J. Luu, I. Kuon, P. Jamieson, T. Campbell, A. Ye, W. M. Fang, and J. Rose, "Vpr 5.0: Fpga cad and architecture exploration tools with single-driver routing, heterogeneity and process scaling," in *ACM FPGA*, 2009.
[23] D. P. Singh and S. D. Brown, "Integrated retiming and placement for field programmable gate arrays," in *ACM FPGA*, 2002.
[24] A. T. Ishii, C. E. Leiserson, and M. C. Papaefthymiou, "Optimizing two-phase, level-clocked circuitry," *J. ACM*, vol. 44, pp. 148–199, January 1997.
[25] B. Lockyear and C. Ebeling, "Optimal retiming of level-clocked circuits using symmetric clock schedules," *IEEE TCAD*, vol. 13, no. 9, sep 1994.