Clustering of Flip-Flops for Useful-Skew Clock Tree Synthesis*

Chuan Yean Tan†, Rickard Ewetz‡, and Cheng-Kok Koh†
†Purdue University, West Lafayette, IN 47906, US
‡University of Central Florida, Orlando, FL 32816, US
e-mail: tan56@purdue.edu, rickard.ewetz@ucf.edu, and chengkok@purdue.edu

ABSTRACT

The clock network of a circuit is a main contributor to the power consumption of any ASIC design. A key technique that is used to reduce power consumption is to cluster flip-flops or latches into groups and to place each group of flip-flops close together to reduce the clock wire length. In this paper, we introduce a clock tree synthesis methodology that incorporates clustering with a previously published useful-skew clock tree synthesis technique to minimize the clock wire length. The clustering process is guided by bounded arrival time constraints, which enable its efficiency. Experimental results show that the proposed methodology reduces up to 34% of the total power consumption while meeting all timing constraints.

I. INTRODUCTION

Power consumption is one of the most important constraints in VLSI design. Low power is required to extend the battery life of mobile devices but also to control the thermal profile of a chip. However, designing a chip that has low power consumption has been challenging and there is typically a trade-off between power consumption and timing performance.

Clock tree synthesis (CTS) deals with constructing a clock tree that delivers a synchronizing clock signal to the sequential elements with minimal clock power consumption. Zero skew trees (ZSTs) [13], bounded-skew trees (BSTs) [7] and useful-skew trees [12, 4] are some of such tree-structures. In particular, USTs utilize useful skew of a given circuit to minimize the wire length and hence, the capacitive cost of the clock tree.

However, in [12] for example, the construction of a UST requires the dynamic incremental updates of skew constraints between all pairs of flip-flops, a time-consuming operation. This problem can be overcome by decoupling the skew constraints between pairs of flip-flops [6, 5]. In these approaches, the skew constraints between pairs of flip-flops are transformed into arrival time ranges for each flip-flop. The immediate effect is that there are only $O(n)$ time constraints to be considered in the synthesis process (instead of $O(n^2)$ all-pairs skew constraints). Moreover, when the arrival time of a flip-flop is optimized or determined in the synthesis process, it takes only $O(1)$ to update its arrival time constraint.

In recent years, clustering of flip-flops has also been proposed to reduce total power consumption. The clustering of a group of flip-flops and relocating them to a central location serves a few purposes. First, it reduces the problem size of clock tree synthesis. Second, each cluster can be driven by a clock buffer, which reduces the overall load that the top-level clock tree (above the leaf-level clusters) has to drive, leading to a more balanced and lower cost tree topology. Several heuristic approaches have been proposed to cluster flip-flops based on both statistical and circuit constraints to minimize clock path wire length. In [15, 10], flip-flops were clustered using the k-means algorithm. The clustering of flip-flops was not guided by timing constraints but by the displacement constraints. While the clusters in [15] were organized in the form of rectangular blocks, the clusters in [10] are transformed into multi-bit flip-flops [3] to further reduce the total power consumption. In [9, 14, 2], the timing slack extracted from post-placement were utilized to compute a feasible merging region for each flip-flop. Similar to [10], the clustered flip-flops in [9, 14, 2] were transformed into multi-bit flip-flops. The clock trees in [15, 10, 14] were constructed using commercial tools; [2] did not specify how the clock trees were constructed.

Our proposed algorithm targets a fast and timing aware clustering technique to reduce total power consumption. The goal is to incorporate flip-flop clustering into a useful-skew clock tree synthesis flow. The timing constraints of a circuit must be met to maintain functional correctness.

While directly using timing constraints to guide the clustering process, our algorithm does not require the construction of a clock tree to derive the necessary timing constraints. We propose a fast and efficient timing aware methodology that clusters flip-flop at a pre-clock tree synthesis stage. Our key contribution can be summarize as below:

1. The algorithm uses arrival time constraints to guide the clustering of flip-flops and displacing them to a new location. The skew constraints of the neighboring flip-flops are updated when a flip-flop is displaced before computations can be made on the other flip-flops to ensure the timing constraints are not violated. Using arrival time constraints simplifies the updating process while guaranteeing timing correctness.

2. As arrival time constraints are being used, the UST synthesis flow from [5] can be directly applied on the clustered solution.

3. The proposed clustering algorithm decrease up to 22% of clock wire capacitance and up to 34% of clock wire capacitance when multi-bit flip-flop transformation is applied. We also show that the clock network wire length decreases as compared to the slight increase in data path wire length.

*This research was supported in part by grant NSF-1527562.

978-1-5090-0602-1/18/$31.00 ©2018 IEEE

II. Preliminaries

A. Timing constraints

Every synchronous circuit is required to satisfy setup time and hold time constraints. There is a setup and hold time constraint imposed between a launching flip-flop and a capturing flip-flop, which are separated by only combinational logic. The setup and hold time constraint between the launching flip-flop FF_i and a capturing flip-flop FF_j:

$$t_i + t_i^{CQ} + t_{ij}^{max} + t_j^S \leq t_j + T, \quad (1)$$

$$t_i + t_i^{CQ} + t_{ij}^{min} \geq t_j + t_j^H, \quad (2)$$

where t_i and t_j are respectively arrival times of the clock signal to FF_i and FF_j; t_{ij}^{min} and t_{ij}^{max} are respectively the minimum and maximum propagation delays through the combinational logic; t_i^{CQ} is the clock-to-output delay of FF_i; T is the clock period; t_j^S and t_j^H are respectively the setup time and hold time of FF_j. The setup and hold time constraint can be reformulated as skew constraints as follows:

$$-l_{ij} \leq t_i - t_j \leq u_{ji}, \quad (3)$$

where $l_{ij} = t_{ij}^{min} + t_i^{CQ} - t_j^H$ and $u_{ji} = T - t_{ij}^{max} - t_i^{CQ} - t_j^S$. The skew constraints in Eq. (3) can captured in a skew constraint graph (SCG). In an SCG $G = (V, E)$, the vertices V represent clock sinks and the edges E represent the skew constraints. For example, the flip-flops FF_i and FF_j are represented as vertex i and j, respectively. Next, an edge e_{ji} is added with a weight $w_{ji} = u_{ji}$ and an edge e_{ij} is added with a weight $w_{ij} = l_{ij}$. We effectively have each edge from j to i representing the constraint

$$t_i - t_j \leq w_{ji}, \quad (4)$$

Based on the properties in the SCG, it can be determined whether a ZST, BST, or UST can satisfy the skew constraints. A ZST can satisfy the skew constraints if all edge weights are non-negative. A BST with a skew bound of B will satisfy the skew constraints if all edge weights are larger than $\frac{B}{2}$. A UST can satisfy the skew constraints if the SCG does not contain any negative cycles.

Displacing a flip-flop changes the weights of edges connecting to the flip-flop in the SCG because the edge weights depend on the propagation delays through the combinational logic, see Eq. (1) and Eq. (2). Therefore, the problem of clustering flip-flops for ZST, BST, or UST construction becomes that of displacing flip-flops while ensuring that the updated SCG has no negative edge weights, no edge weight smaller than $\frac{B}{2}$, or no negative cycles, respectively.

B. Displacing a flip-flop using a feasible region

Many timing-aware techniques of clustering flip-flops are based on the techniques proposed in [9]. In [9], a technique is proposed to move (or displace) a single flip-flop without introducing any negative edge weights in the SCG for a ZST. Note that the algorithm can be generalized to handle BST construction with a skew bound B by decrementing each edge weight with $\frac{B}{2}$ prior to the optimization.

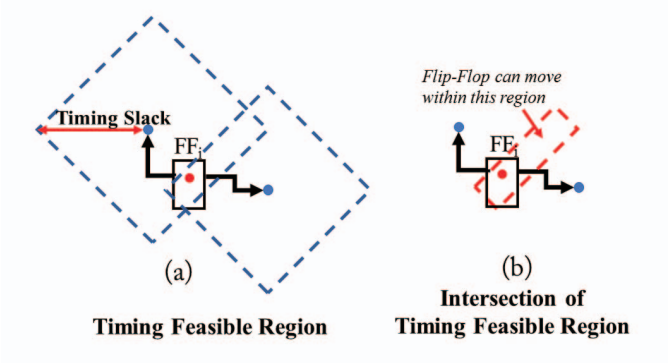

Fig. 1. (a) Tilted squares derived from timing slacks based on fanin and fanout gates. (b) Feasible region of flip-flop defined by intersection of tilted squares in (a).

When a flip-flop FF_k was displaced in [2], it was observed that the propagation delays through the combinational logic in the fanin and fanout of FF_k may change. Consequently, the corresponding edge weights are required to be updated in the SCG. Moreover, a feasible region (FR) was introduced such that if FF_k is placed within FR_k, it is ensured that every edge weight in the SCG is non-negative.

The FR is first determined by forming tilted squares as shown in Figure 1(a) at each of the combinational fanin and fanout gates of the the corresponding flip-flop. Next in Figure 1(b), the FR is defined to be the intersection of these tilted squares. The size of each tilted squares is computed using the Elmore delay model and the slack on the timing path through each of the combinational gates, i.e., the edge weights in the SCG (see details in [9]). Next, we outline how the FR are used to guide the clustering of flip-flops.

C. Timing-driven clustering of flip-flops in previous studies

A number of different timing-driven clustering methodologies have been introduced based on the FR presented in [2]. The main idea is to form the FR for each flip-flop. If all the FRs of a group of flip-flops intersect, these flip-flops can be clustered. In the methods of [9, 14, 2], they all relied on finding on the maximal cliques of flip-flops with intersecting FRs.

However, a flip-flop may participate in many different groups but it can only be clustered with one of these groups. The selection of the group to which a flip-flop belonged was performed based on different optimization objectives. In [2], the selection is based on finding a maximum independent set using a heuristic approach. In [9], maximal cliques are found using x- and y-interval graphs. Moreover, they identified "decision points" at which essential flip-flops are to be clustered. Maximum clique of flip-flops with intersecting FRs was found. In [14], a sampling technique is used to cluster flip-flops.

D. Dynamic update on clustered flip-flop

When clustering a group of flip-flops, for each flip-flop that is displaced, their respective FRs may have to be updated since the constraints among these flip-flops may not be independent. For example, consider the clustering of two flip-flops FF_i and

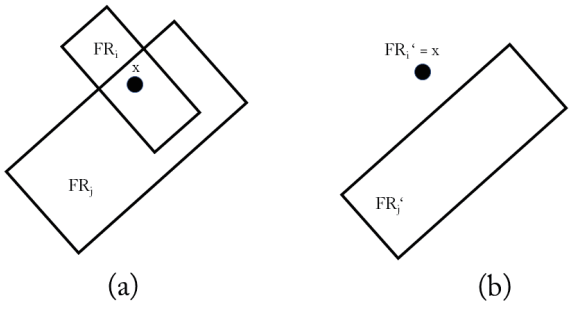

Fig. 2. (a) Two intersecting FRs: FR_i and FR_j. (b) After moving FF_i to location x, $FR_i' = x$, and FR_j' should be modified, and it may not contain x.

FF_j with overlapping FRs FR_i and FR_j as in Figure 2(a). After FF_i is moved to a location x, the edges in the fanin and fanout of FF_i must be updated. If this is the final location of FF_i, we can treat that the new FR of FF_i, denoted as FR_i', is essentially x. Moreover, the weights of edges that represent the connections between FF_i and the other flip-flops in the circuit must be updated in the SCG. If FF_j is a fanin or fanout of FF_j, the FR of FF_j must also be updated. i.e., the FR of FF_j. In Figure 2(b), it can be observed that the updated FR of FF_j, denoted as FR_j', does not include the location x. Consequently, if the two FFs are directly merged at location x, as in some previous studies, the timing constrains would be violated.

In addition, the earlier studies assumed zero skew or bounded skew constraints. However, for modern designs with stringent performance requirement, it may be necessary to incorporate useful skew in the design. When useful skew is needed, flip-flops are allowed to be moved as long as no negative cycles are formed in the SCG. Hence, certain edges in the SCG can be allowed to be negative. However, forming an FR while ensuring no negative cycles are created would require a run-time complexity of $O(V \log V + E)$ [4], which would translate into prohibitively long run-times.

To summarize, the main challenges of flip-flop clustering are: (1)The feasible regions of different flips flops are not independent. (2)The previous approaches do not facilitate the use of useful skew. We will address these two main challenges in this paper. In particular, the clustering problem and the useful-skew tree synthesis problem are defined as follows: Given an arbitrary circuit with flip-flops, cluster as many flip-flops together given their physical location and skew constraints, and synthesize a useful-skew clock tree for the clustered flip-flops, with the goal of minimizing the total power consumption, while meeting all skew constraints.

III. Proposed clustering technique

In this section, the proposed approach of clustering flip-flops is presented. The proposed technique allows the use of useful skew and decouples the movement of every flip-flop. The proposed approach is based on the bounded useful arrival time constraints proposed in [1], which are used the clock tree synthesis algorithms in [6, 5]. Therefore, we first introduce the bounded useful arrival time constraints in Section A before we present the proposed clustering technique in Section B.

A. Arrival time constraints

Based on the skew constraints captured in the SCG, a set of bounded useful arrival time constraints can be defined. The constraints consist of an arrival time range for each flip-flop, which is illustrated in Figure 3(a). Let the arrival time range for FF_i be denoted $[x_i^{lb}, x_i^{ub}]$, where x_i^{lb} and x_i^{ub} are the lower and upper bound of the range respectively. The arrival time ranges should be defined such that if the clock signal is delivered to each flip-flop within its respective arrival time range, all the skew constraints in the SCG will be satisfied.

A set of arrival time ranges exist if there is no negative cycle in the SCG. In this paper, the arrival time range for each flip-flop is computed using the LP formulation in [5], as follows:

$$\min \sum_{i \in V} f^{lb}(x_i^{lb}) + f^{ub}(x_i^{ub}) \tag{5}$$

$$x_i^{lb} \leq x_j^{ub} \tag{6}$$

$$x_i^{ub} - x_j^{lb} \leq w_{ji} \tag{7}$$

where $f^{lb}(x)$ and $f^{ub}(x)$ are two piece-wise linear functions that attempt to specify the constraints to be relatively aligned and have long ranges. The details of the objective function are explained in [5]. Eq. (6) ensures that the lower bound is smaller than the upper bound. Eq. (7) is used to ensure that the skew constraints in the SCG, i.e., Eq. (4), are satisfied and therefore, no negative cycles are formed in the SCG.

The advantage of using arrival time constraints to guide the flip-flop merging (or clock tree synthesis) is that the arrival time constraints are decoupled. Therefore, it is possible to evaluate the constraints in constant time. In [5, 6], the constraints were used to allow the exploration of various tree topologies in the tree construction process.

We propose to use the bounded useful arrival time constraints to guide the clustering optimization as the constraints allow the use of useful skew and the arrival time ranges for the different flip-flops are decoupled from each other.

B. Clustering operations

Our proposed algorithm of clustering flip-flop is based on using the bounded useful arrival time ranges introduced in the previous section. In particular, we introduce a flip-flop matching operation and an arrival time range scaling operation. In Section IV, we present a framework where flip-flops are clustered using the two operations.

The flip-flop matching operation is used to evaluate whether a flip-flop pair has matching arrival time constraints and whether they can be relocated to be placed close to each other. Note that when they are placed close to each other, we can apply further optimization and transform the clustered flip-flops into a multi-bit flip-flops (MBFF) [3]. For example, when flip-flop FF_i and flip-flop FF_j are placed at the same location and clustered in flip-flop cluster k, we define an arrival time range for the cluster k, denoted as $[x_k^{lb}, x_k^{ub}]$, as follows:

$$[x_k^{ub}, x_k^{lb}] = [\min\{x_i^{ub},\ x_j^{ub}\}, \max\{x_i^{lb},\ x_j^{lb}\}], \tag{8}$$

where the arrival time ranges are illustrated in Figure 3(a) and Figure 3(b). Note that if no intersection exists between the arrival time ranges of FF_i and FF_j, the two flip-flops cannot be

merged. Moreover, even if we do not merge the flip-flop pair, we would not want to place flip-flops with non-intersecting arrival times close to each other because they cannot be directly joined in the clock tree. Moreover, it is difficult to implement a clock tree that delivers the clock signal that has a small arrival time range. Therefore, to ease the synthesis of the clock tree, we impose a constraint on the minimum range $r_k = x_k^{ub} - x_k^{lb}$ of an arrival time range, i.e., $r_k > r_{user}, \forall k$, where r_{user} is a user-defined parameter.

The arrival time range scaling operation is used to check if two flip-flops with matching arrival time ranges can be displaced to the same location. Displacing flip-flop incurs timing delays from additional wire length introduced. If a flip-flop i is displaced to a location to form cluster k, wire delays are accounted for by scaling the length of the arrival time as follows:

$$[\hat{x}_k^{lb}, \hat{x}_k^{ub}] = [x_k^{lb} + \triangle_k, x_k^{ub} - \triangle_k], \tag{9}$$

where $\triangle_k$ is the sum of the worst-case changes in delay (after the displacement) from any of the fanin or fanout gates to flip-flops in cluster k. Figure-3(c) illustrates a pair of candidate flip-flops to be clustered where the locations of its fanin and fanout gates are fixed. When the flip-flops are clustered and displaced to a central location, the data path wire length will increase in the worst case. For each flip-flop FF_i in the cluster, the change in the delay at the fanin can be estimated as follows:

$$\begin{aligned}\triangle &= (R_{driver} + R_{wire} \cdot l_{wire}) \cdot C_{wire} \cdot \triangle(x,y) \\ &+ R_{wire} \cdot \triangle(x,y) \cdot (C_{wire} \cdot \triangle(x,y)/2 + C_{FF}),\end{aligned} \tag{10}$$

where R_{driver} is the resistance of the fanin gate of FF_i, R_{wire} and C_{wire} are respectively the per-unit length wire resistance and capacitance, l_{wire} is the original wire length, $\triangle(x,y)$ is the change in wire length, and C_{FF} is the input load capacitance of the flip-flop. The change in the delay at the fanout can be estimated in a similar fashion. $\triangle_k$ is obtained by summing up all such delay changes of all flip-flops in cluster k, as shown in Figure 3(d), which assumes that the clustered flip-flops are transformed into a MBFF. Note that the computation of $\triangle_k$ is based on a pessimistic estimation of how the displacement of a flip-flop will affect the timing. While it reduces the timing range, it also ensures that the timing constraints will always be met after clustering.

If the matching and scaling operations do not result in an empty arrival time range, the constraints specified in Eq. (4) hold as follows. Suppose FF_i is selected to be clustered (into cluster k) and FF_i is initially displaced to a new location. The following inequality can be formed:

$$x_i^{lb} \le \hat{x}_k^{lb} \le \hat{x}_k^{ub} \le x_i^{ub}. \tag{11}$$

For any flip-flop FF_j that is connected to FF_i in the SCG, it is easy to verify that

$$\hat{x}_k^{ub} - x_j^{lb} \le x_i^{ub} - x_j^{lb}. \tag{12}$$

In other words, the inequality in Eq. (4) still holds. We can make the same argument for other flip-flops in cluster k. This is also true when the cluster k contains flip-flops that are connected in the SCG.

It should be clear that using the two operations, it can be checked efficiently whether a pair of flip-flops can be clustered (when the matching and scaling operations do not produce an empty arrival time range). In the next section, we present the overall framework for flip-flop clustering and UST synthesis.

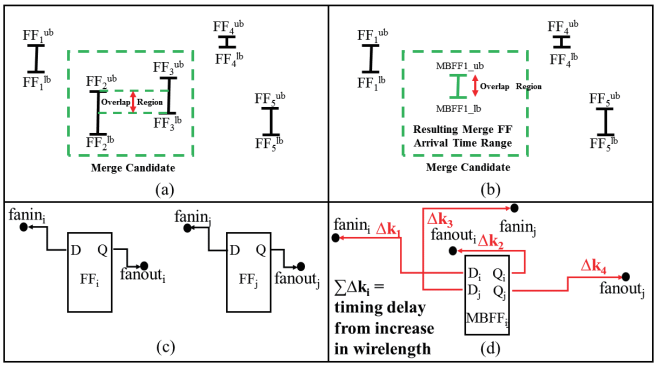

Fig. 3. (a) Arrival time range overlap comparison; (b) Resulting arrival time range from matching; (c) Original circuit before clustering (d) $\triangle_k = \sum \triangle_{k_i}$, where $\triangle_{k_i}$ are delay changes from flip-flops in cluster k that are used to scale arrival time ranges.

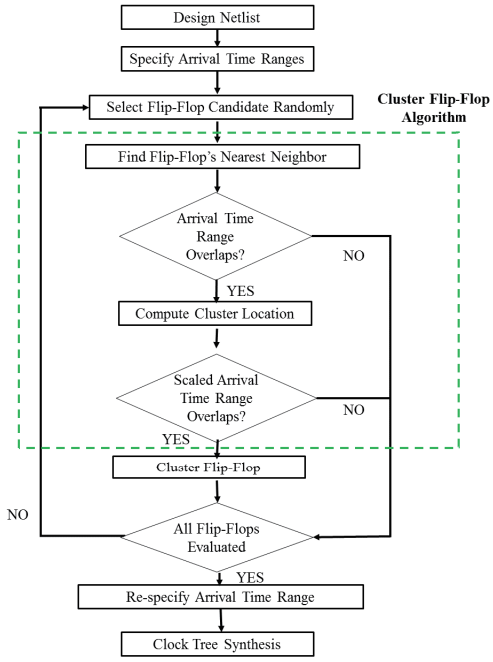

Fig. 4. Flow of Proposed Methodology

IV. Proposed Methodology

Figure 4 illustrates an overview of our proposed clustering and synthesis methodology. Given the skew constraints and physical locations of flip-flops, the methodology first determines the arrival time ranges. This is obtained by solving the linear program Eq. (5–7) in Section A. Based on the specified arrival time ranges, clustering is then performed (see Section A), and the arrival time ranges of the clustered flip-flops are re-specified (see Section B). The re-specified arrival time ranges are then used for the synthesis of a useful-skew tree for the clustered flip-flops (see Section C).

A. Clustering and displacement of flip-flops

We use a greedy approach to perform clustering of flip-flops. First, we randomly select a flip-flop and compute its nearest neighbor. Nearest neighbors are considered for clustering because we want to minimize the total displacement of each flip-flop for two reasons: (i) To minimize the increase in wire length of the data path; and (ii) To minimize the delay changes of the fanin and fanout gates associated with the flip-flop. We now compare their respective arrival time ranges (see Eq. (8) for flip-flop matching) to decide whether to cluster the pair.

After determining that the flip-flops are candidates for clustering, we compute the potential cluster location for the flip-flops. We use a weighted approach to determine the new location at which these flip-flops would be clustered. Each flip-flop has a different number of fanout gates. The fanout of a flip-flop translate to the amount of capacitive load seen by each flip-flop. Hence, displacing a flip-flop that has higher capacitive load would incur a higher delay change. Since we want to minimize the amount of timing delay caused by the displacements, the capacitive load of a flip-flop is used to influence the determination of the cluster location.

B. Scaling arrival time ranges

Once the new location is determined for the cluster, the Elmore delay model is used to compute the delay change at the fanin and fanout of the respective flip-flops. $\triangle_k$, the sum of these delay changes stemming from the displacement of flip-flops is used to tighten the arrival time range of the cluster using Eq. (9). If the resulting arrival time range is non-empty, the candidate flip-flops are clustered and locked. They are not considered for further clustering in this iteration. Otherwise, the flip-flops can be considered for clustering with other flip-flops within this iteration.

An iteration completes when all flip-flops are either locked or cannot be clustered with other flip-flops. We can re-iterate the process until no further clustering is possible.

C. Synthesizing useful-skew tree

In our methodology, we use a useful-skew tree construction technique based on arrival time ranges for clock tree synthesis [5]. After the clustering process, we have a new set of flip-flop, clustered flip-flop locations and their corresponding arrival time ranges. In the clock tree synthesis step, we do not physically synthesize the layout of a flip-flop cluster. We assume that the cluster is represented as a single point in a 2-D space (as in the case of a flip-flop).

We also consider the transformation of a flip-flop cluster to a multi-bit flip-flop before clock tree synthesis. This transformation can further decrease the power consumption of each flip-flop through the shared clock buffer in the multi-bit flip-flop cell. In this case, the clock pin capacitance are scaled appropriately after the multi-bit flip-flop transformation before we apply clock tree synthesis.

V. Results and Evaluation

In this section, we will discuss the experiments and results in detail. Our algorithm is implemented in C++ and the experimental results are performed on a 10 core 5.0GHz Linux machine with 64GB of memory. We have run and evaluated our proposed methodology on the open-source IWLS 2005 benchmark circuits [8]. These benchmark circuits vary in both design size as shown in Table I.

TABLE I
IWLS Benchmark Design Circuits.

No.	Design Name	# flip-flops
1	USB Fast	1752
2	Direct Memory Access	2121
3	openMSP430	686
4	Ethernet	10544
5	AES 256	13216
6	PCI Bridge	3582
7	VGA enhanced	17071

TABLE II
Comparisons of Methodologies.

Design	Methodology	Min. Slack (ps)	Clock Cap (pF)	Sink Cap (pF)	Total Cap (pF)	Exec. Time (min)
1	ZST [13]	57	4.58	1.04	5.62	0.5
	Cluster+ZST	59	4.04	1.04	5.08	0.4
	UST [5]	34	2.49	1.04	3.53	3.6
	Cluster+UST	52	2.16	1.04	3.20	1.0
	MBFF+UST	41	1.93	0.55	2.48	0.8
2	ZST [13]	90	5.66	1.28	6.94	0.9
	Cluster+ZST	84	4.28	1.28	5.56	0.5
	UST [5]	84	3.03	1.28	4.31	2.1
	Cluster+UST	54	2.44	1.28	3.72	0.9
	MBFF+UST	28	2.20	0.65	2.85	1.2
3	ZST [13]	120	2.28	0.38	2.66	0.4
	Cluster+ZST	119	1.60	0.38	1.98	0.2
	UST [5]	120	1.06	0.38	1.44	0.8
	Cluster+UST	119	0.85	0.38	1.23	0.4
	MBFF+UST	119	0.90	0.20	1.10	0.5
4	ZST [13]	62	25.44	6.23	31.67	15.4
	Cluster+ZST	60	20.00	6.23	26.23	44.6
	UST [5]	19	16.31	6.23	22.54	13.8
	Cluster+UST	14	12.68	6.23	18.91	44.0
	MBFF+UST	14	11.01	3.24	14.25	47.7
5	ZST [13]	51	33.57	6.90	40.47	19.9
	Cluster+ZST	59	26.85	6.90	33.75	6.2
	UST [5]	19	18.19	6.90	25.09	21.2
	Cluster+UST	24	15.82	6.90	22.72	9.8
	MBFF+UST	12	14.26	3.54	17.80	9.8
6	ZST [13]	70	8.39	2.16	10.55	2.1
	Cluster+ZST	70	6.99	2.16	9.15	0.9
	UST [5]	23	5.32	2.16	7.48	3.3
	Cluster+UST	33	4.36	2.16	6.52	1.6
	MBFF+UST	50	3.88	1.12	5.00	2.0
7	ZST [13]	59	39.59	10.30	49.89	97.4
	Cluster+ZST	59	30.78	10.30	41.08	18.7
	UST [5]	59	24.53	10.30	34.83	25.5
	Cluster+UST	75	19.73	10.30	30.03	9.6
	MBFF+UST	36	17.66	5.26	22.92	12.2

For each of the benchmark circuits in Table I, we perform clustering followed by useful-skew tree (UST) construction. The characteristics of such trees are provided in Table II in

TABLE III
REDUCTION IN CLOCK PATH WIRING COST VERSUS INCREASE IN DATA PATH WIRING COST AFTER CLUSTERING.

Design	Data Path Wire Cap Reduction (%)	Clock Path Wire Cap Reduction (%)
1	-4.17	13.25
2	-9.86	19.47
3	-7.29	19.81
4	-6.02	22.25
5	-1.80	13.03
6	-4.81	18.05
7	-8.98	19.56

the rows labeled "Cluster+UST". In both case, the USTs are constructed using the method in [5].

We compare the performances of topologies "Cluster+UST" and "MBFF+UST" against zero-skew trees (ZST) [13] and UST [5]. We also compare against ZSTs constructed on clustered flip-flops. The results of these different methods are provided in Table II in the rows labeled "ZST", "UST", and "Cluster+ZST", respectively.

In Table II, the clock capacitance ("Clock Cap") is the sum of the wire and buffer capacitance in the clock tree. The total clock capacitance ("Total Cap") comprises clock capacitance, and the clock pin capacitance of flip-flops. The benefits of flip-flop clustering and UST construction for clustered flip-flops are clear from Table II. There is a 8% to 14% decrease in total capacitance where the main savings were contributed by the decrase in clock path wire length. The lower the total capacitance, the lower the total power consumption of the design.

We also apply multi-bit flip-flop transformation to the clustered flip-flops, followed by UST construction. The characteristics of such trees are provided in Table II in the rows labeled "MBFF+UST". The results show that the additional power reduction by merging the clustered flip-flops into multi-bit flip-flops. There is a 20% to 34% of power reduction coming from transforming the clustered single bit flip-flops into multi-bit flip-flops. The main savings from utilizing multi-bit flip-flop is the sharing of the internal clock buffer between the flip-flops. Table IV shows the normalized savings of a multi-bit flip-flop [3]. In addition, the input capacitance seen by the clock pin reduces by a factor of 2 for a 2-bit flip-flop. The reduction in the clock input capacitance yields two main benefits: (1) Smaller buffers can be used to drive the multi-bit flip-flops, and (2) fewer buffers are required in the clock tree synthesized.

TABLE IV
MBFF POWER AND AREA SAVINGS [3].

No. of Bits	1-Bit FF	2-Bit FF	4-Bit FF
Power Per Bit	1.00	0.86	0.78
Area Per Bit	1.00	0.96	0.71

The run-times of the clustering and synthesis methodologies are also listed in Table II. In general, clustering should reduce the run-times of the synthesis process. Note that the increase in run-time on circuit 4 is compensated with large reductions in terms of capacitance.

Table II also shows the minimum slack ("Min Slack") of the various topologies under process, voltage, and temperature variations. To do that, we perform Monte Carlo simulations to evaluate the robustness of the topologies synthesized. The results were sampled across 100 Monte Carlo simulations. In each iteration, the following parameters are subject to variations: 1) Wire Width ($\pm 5\%$); 2) Supply Voltage ($\pm 7.5\%$); 3) Temperature ($\pm 15.0\%$); 4) Channel Length ($\pm 5.0\%$). The minimum slack is the smallest among these simulations for each topology. It is clear that clustering does not degrade the robustness of the clock tree topologies.

The changes in data path wire capacitance are also investigated to understand the effects of flip-flop clustering. For fanin, we assume a simple two-pin net connection. For the connections of a flip-flop to the fanout gates, we construct a Steiner tree before clustering and construct another Steiner tree for comparison after clustering (and displacement). (Note that for the worst case delay change, we assume that the overhead is that of an additional wire connecting the root of the original Steiner tree to the new location of the cluster.) From Table III, the clock path wire capacitance reduces by 13%-22% while the data path wire capacitance increases by 1%-10%.

VI. SUMMARY

In summary, we presented a flip-flop clustering algorithm based on arrival time ranges derived from clock skew constraints. We also constructed useful-skew trees on the clustered flip-flops. Experimental results showed significant reduction in clock capacitance.

REFERENCES

[1] C. Albrecht, B. Korte, J. Schietke, and J. Vygen. Maximum mean weight cycle in a digraph and minimizing cycle time of a logic chip. *Discrete Applied Mathemathics*, 123(1-3):103–127, 2002.

[2] Y. T. Chang, C. C. Hsu, M. P. H. Lin, Y. W. Tsai, and S. F. Chen. Post-placement power optimization with multi-bit flip-flops. In *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 218–223, Nov 2010.

[3] L. Chen, A. Hung, H. Chen, E. Tsai, S. Chen, M. Ku, and C. Chen. Using multi-bit flip-flop for clock power saving by designcompiler. *Proceedings of Synopsys Users Group*, 2010.

[4] R. Ewetz, S. Janarthanan, and C. K. Koh. Fast clock skew scheduling based on sparse-graph algorithms. In *The 20th Asia and South Pacific Design Automation Conference*, pages 472–477, Jan 2015.

[5] R. Ewetz and C.-K. Koh. Clock tree construction based on arrival time constraints. In *Proceedings of the 2017 ACM on ISPD*, ISPD '17, pages 67–74, New York, NY, USA, 2017. ACM.

[6] S. Held, B. Korte, J. Massberg, M. Ringe, and J. Vygen. Clock scheduling and clock-tree construction for high performance asics. ICCAD'03, pages 232–239, 2003.

[7] D. J.-H. Huang, A. B. Kahng, and C.-W. A. Tsao. On the bounded-skew clock and steiner routing problems. DAC'95, pages 508–513, 1995.

[8] International Workshop for Logic Synthesis. http://irwls.org/iwls2005/benchmarks.html. 2005.

[9] I. H. R. Jiang, C. L. Chang, and Y. M. Yang. INTEGRA: Fast multibit flip-flop clustering for clock power saving. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(2):192–204, Feb 2012.

[10] A. B. Kahng, J. Li, and L. Wang. Improved flop tray-based design implementation for power reduction. In *Proceedings of the 35th International Conference on Computer-Aided Design*, page 20. ACM, 2016.

[11] M. P. H. Lin, C. C. Hsu, and Y. C. Chen. Clock-tree aware multibit flip-flop generation during placement for power optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(2):280–292, Feb 2015.

[12] C. W. A. Tsao and C. K. Koh. UST/DME: A clock tree router for general skew constraints. In *IEEE/ACM International Conference on Computer Aided Design. ICCAD - 2000. IEEE/ACM Digest of Technical Papers (Cat. No.00CH37140)*, pages 400–405, Nov 2000.

[13] R.-S. Tsay. Exact zero skew. ICCAD'91, pages 336–339, 1991.

[14] S. H. Wang, Y. Y. Liang, T. Y. Kuo, and W. K. Mak. Power-driven flip-flop merging and relocation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(2):180–191, Feb 2012.

[15] G. Wu, Y. Xu, D. Wu, M. Ragupathy, Y. Y. Mo, and C. Chu. Flip-flop clustering by weighted k-means algorithm. In *2016 53nd ACM/EDAC/IEEE (DAC)*, pages 1–6, June 2016.