

Optimal Prescribed-Domain Clock Skew Scheduling

Li Li, Yinghai Lu, Hai Zhou
Electrical Engineering and Computer Science
Northwestern University

Abstract—Clock skew scheduling is an efficient technique to minimize the cycle period by properly assigning clock delays to registers in a circuit. But its effectiveness is limited by the difficulty in implementing a large number of arbitrary clock skews. Multi-domain clock skew scheduling and prescribed-domain clock skew scheduling are two alternatives to overcome this shortage by restricting the number of clock domains. While multi-domain clock skew scheduling has been proved to be NP-hard, the hardness of prescribed-domain clock skew scheduling algorithm remains evasive. In this paper, we give a positive answer to the open question by presenting the first efficient and optimal algorithm for prescribed-domain clock skew scheduling. Besides the runtime improvement over the previous method, the experimental results on ISCAS89 benchmarks show comparable quality to those generated by optimal multi-domain clock skew scheduling.

I. INTRODUCTION

Clock skews are differences in clock arrival times at different registers due to the differences in interconnect delays in the clock distribution network. In the early stage of sequential circuit design, it used to be considered as a hazard because it may complicate the static timing constraints and cause design difficulties. To eliminate the effect of clock skews, a series of work [1], [2], [3], [4], [5] have been dedicated to build the zero-skew clock trees. However, it is soon found [6] that the performance of the circuits can be well improved by properly assigning clock delays to different registers. This technique, which is also referred as *clock skew scheduling* (CSS), has been widely applied in various circuit optimization problems [7], [8], [9], [10], [11] in the last two decades.

Although clock skew scheduling appears to be very efficient in minimizing clock periods, its application in practical design is compromised by the arbitrary skew distribution in the optimal solution [10]. Due to process variation, it is unrealistic to implement clock schedule with a large set of arbitrary delays. In order to deal with the difficulty of unconstrained clock scheduling, Ravindran et al. [10] proposed the concept of multi-domain clock skew scheduling (MDCSS), in which the clock phase shifts of all registers are constrained to a small sets called clock domains. Under a user given number of clock domains, MDCSS will compute the optimal clock phase shift for each domain and assign registers to domains to minimize the clock period. In [10], the problem is formulated as a mix integer linear programming (MILP) and solved by SAT-solver-based branch-and-bound method. Though their solution achieves optimality, the SAT-solver-based approach generally encounters high computational overhead.

After the pioneering work of [10], various research works have been dedicated to the MDCSS problem. Casanova et

al. [12] proposed a multi-level clustering algorithm to tackle this problem. The algorithm recursively groups the registers according to their skew affinity, which is estimated by the overlapping of their slack intervals. Their heuristic turns out to be much faster than [10], but also shows degradation in solution quality. Zhi et al. [13] presented a practical framework combining branch-and-bound and clustering. Though their algorithm obtains optimal results on tested benchmarks within a short time, there is still no guarantee on the performance of branch-and-bound. In [14], Zhi et al. further formulated the MDCSS problem as a special MILP, which can be efficiently solved by generalized Howard's algorithm for MCR (minimum cycle ratio) problem. Ni et al. [15] studied the problem from a different perspective. They argued that minimizing the domain number will be better in terms of achieving the best clock period. However, the gain on clock period may not be able to compensate the high cost on implementing clock trees since the resulted domain number is unpredictable. There are also other works that take advantage of MDCSS to optimize different objectives, such as [8] that considers peak current and [16] that considers clock distribution network.

Despite of all the efforts dedicated to the multi-domain clock skew scheduling problem, it is still hard to find a solution with a satisfiable trade-off between the runtime and optimality, as recently the MDCSS problem has been proved to be NP-hard [17].

An alternative approach to overcome the domain explosion in clock skew scheduling is prescribed-domain clock skew scheduling (PDCSS), which is first proposed by Lin et al. [7]. In PDCSS, the number of clock domains are defined by users as well as in MDCSS. However, unlike MDCSS which will automatically generate the domain skew (clock phase shift) for each clock domain, PDCSS accepts a set of user-defined domain skews that is proportional to cycle period. In [7], the authors attack the PDCSS problem indirectly and provide a two-step solution. First, an optimal cycle period is obtained by considering setup constraints only. Then the violated hold constraints are fixed by delay padding [18]. Though their results appear to be competitive, technically speaking, the method in [7] did not exactly solve the PDCSS problem, as it turns out that delay-padding dominates the overall runtime and incurs extra area and power cost.

PDCSS gives the designers better control on the clock domain distribution with the price of cycle period, thus it is a preferable choice when there are special constraints on clock trees and clock paths. More importantly, we find that PDCSS has an efficient and effective solution while MDCSS has no guarantee on the solution due to its NP-completeness.

In this paper, we push the progress of research on PDCSS by presenting a polynomial-time optimal algorithm for the first time. Our approach directly aims at the PDCSS problem by considering setup and hold constraints simultaneously. Compared to the existing PDCSS algorithm, it yields the optimal cycle period and does not incur other cost. Furthermore, the experiment shows that the cycle periods generated by our PDCSS algorithm are competitive to the optimal cycle periods generated by previous MDCSS algorithms while our runtime is less.

The rest of the paper is organized as follows. Section II presents the problem formulation and some characteristics of this problem. Our algorithm is elaborated in Section III, where its property is also discussed. Experimental results are presented in Section IV, followed by conclusions in Section V.

II. PROBLEM FORMULATION AND CHARACTERISTICS

In this section, we give the formulation of PDCSS and point out its special problem structure which makes it difficult to solve.

A. Problem Formulation

Let $G = (V, E)$ denote the timing graph for a circuit, where each node $v \in V$ represents a register with a clock skew $l(v)$, and let $d_{\text{setup}}(v)$ and $d_{\text{hold}}(v)$ denote the setup time and hold time of register v . E contains a directed edge (u, v) if there is any combinational path from register u to register v . The longest and shortest combinational path delays are denoted by $D(u, v)$ and $d(u, v)$, respectively. Furthermore, let T denote the clock period. In the following discussion, we will assume that $D(u, v)$, $d(u, v)$, $d_{\text{setup}}(v)$, $d_{\text{hold}}(v)$ are known constants.

For each $(u, v) \in E$, a feasible clock skew of u and v must satisfy the following difference inequalities to meet the timing constraints of circuit:

$$l(u) \leq l(v) + T - D(u, v) - d_{\text{setup}}(v), \quad (1)$$

$$l(u) \geq l(v) + d_{\text{hold}}(v) - d(u, v), \quad (2)$$

Inequality (1) is the setup time constraints that guarantee the signal from u to v has been stabilized before register v stores it. Inequality (2) is the hold time constraints that guarantee the signal from u will not overwrite the signal in last cycle before v stores it correctly.

Inequalities (1) (2) can be further translated into graphical language. We can associate a weight $w(u, v) = T - D(u, v) - d_{\text{setup}}(v)$ with edge $(u, v) \in E$ to represent setup constraints. To represent hold constraints, we can introduce a reverse edge (v, u) with weight $w(v, u) = d(u, v) - d_{\text{hold}}(v)$. The resulted timing graph is denoted by G' . For unconstrained clock skew scheduling with arbitrary clock skews, a feasible T that satisfies Inequalities (1) (2) will not produce any negative cycle in G' . And computing the optimal cycle period T_{opt} is equal to finding the MMC (minimum mean cycle) in the weighted cyclic graph.

The clock skew scheduling problem with given number of domains and constant domain skews (CDCSS, Constant-domain clock skew scheduling) has been studied in [19] as

a system of difference constraints with variable constrained to a finite set, which can be optimally solved by a modified Bellman-Ford algorithm. It is worth noting that the CDCSS problem is more like a discrete version of unconstrained clock skew scheduling in nature, rather than related to either PDCSS or MDCSS. Interested readers could refer to [19] for more details.

In our formulation, the domain skews are not constants but proportional to the cycle period. This assumption is reasonable because the clock skews will typically increase when the size of circuit grows. Suppose we are given k skew domains $l_1T, l_2T, \dots, l_kT$ with respect to the clock period T , where $l_1, \dots, l_k$ are constants between 0 and 1 that $l_1 = 0 < l_2 < \dots < l_k < 1$. We use $s(v)$ to denote the choice of skew for register v where $s(v) = j$ if $l(v) = l_jT$. Using these notations and Constraints (1) (2), the Prescribed-domain Clock Skew Scheduling (PDCSS) problem can be formulated as follows:

$$\begin{aligned} \min T \\ \text{s.t. } l_{s(v)} - l_{s(u)} &\geq \frac{D(u, v) + d_{\text{setup}}(v)}{T} - 1, \forall (u, v) \in E \\ l_{s(v)} - l_{s(u)} &\leq \frac{d(u, v) - d_{\text{hold}}(v)}{T}, \quad \forall (u, v) \in E \\ s(v) &\in \{1, \dots, k\}, \quad \forall v \in V \end{aligned} \quad (3)$$

Problem (3) without hold constraints has been studied by Lin et al. [7] and solved by an incremental algorithm. Actually, Problem (3) is very different from either CDCSS or its simplified version without hold constraints because it is non-monotone with respect to cycle period T , i.e. a cycle period T is feasible does not indicate that all cycle periods larger than T is feasible.

B. Non-Monotonicity of Problem

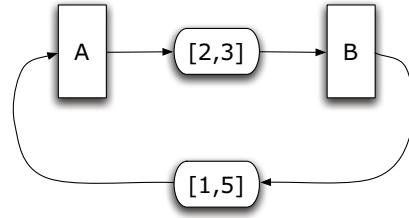


Fig. 1. An example of non-monotone prescribed-domain clock scheduling

We use Figure 1 as an example to illustrate the non-monotone characteristics of the PDCSS problem. In this example, we have two registers A and B with clock skew $l(A)$ and $l(B)$, respectively. The blocks between registers represent the minimum and maximum delays on the combinational paths. For simplicity, suppose the setup time and hold time are both zero. There are two available domain skews, 0 or $1/4$ of the cycle period, i.e. $l_1 = 0, l_2 = 1/4$. Under a given clock period T , $l(A)$ and $l(B)$ must satisfy the following inequalities to meet the setup and hold timing constraints on path $A \rightarrow B$

$$\begin{aligned} l(B) - l(A) &\geq 3 - T \\ l(B) - l(A) &\leq 2 \end{aligned}$$

Similarly, path $B \rightarrow A$ requires the setup and hold timing constraints

$$\begin{aligned} l(A) - l(B) &\geq 5 - T \\ l(A) - l(B) &\leq 1 \end{aligned}$$

In this circuit, the minimum cycle period is 4. When $T = 4$, we can assign $l(A) = T/4 = 1$ and $l(B) = 0$. When $T \geq 5$, $l(A) = l(B) = 0$ will be a feasible assignment. However, no feasible assignment can be found with the clock skews constrained to $\{0, T/4\}$ when $4 < T < 5$. When $4 < T < 5$, there must be $l(A) = T/4 > l(B) = 0$ in order to meet setup constraint on path $B \rightarrow A$. As a result, the hold constraint on path $B \rightarrow A$ is destined to be violated since $l(A) - l(B) = T/4 > 1$.

The non-monotonicity of Problem (3) makes it difficult to find an optimal solution. Although the feasibility of the problem with a given T can be easily checked in polynomial-time [19], binary search approach cannot be applied to find the exact solution as for the PDCSS problem. However, after careful analysis we find that the optimal T must reside in a polynomial-sized set of discrete points, despite of the non-monotonicity of the solution space. In the next section we will show how to reduce the solution space.

III. ALGORITHM

In Problem (3), the lower bound of T is determined by the setup constraints and the upper bound of T is determined by the hold constraints. We naturally observe that the minimum cycle period T_{opt} with prescribed domains is bounded by the setup constraints, which can be stated as the following lemma:

Lemma 1: There exists an edge $(u, v) \in E$ such that

$$T_{opt} = \frac{D(u, v) + d_{setup}(v)}{1 + l_{s(v)} - l_{s(u)}}. \quad (4)$$

Proof: We prove this by contradiction. Given T_{opt} , if there does not exist a tight bound in all setup constraints, i.e. $l_{s(v)} - l_{s(u)} > \frac{D(u, v) + d_{setup}(v)}{T_{opt}} - 1$ holds for every edge $(u, v) \in E$, then we have $T_{opt} > T_{max} = \max_{(u, v) \in E} \frac{D(u, v) + d_{setup}(v)}{1 + l_{s(v)} - l_{s(u)}}$. Next, we use T_{max} as the cycle period and keep all clock skews l unchanged. Since $T_{max} < T_{opt}$ and T_{opt} satisfies all hold constraints, T_{max} satisfies all hold constraints. In addition, T_{max} trivially satisfies all setup constraints because $T_{max} \geq \frac{D(u, v) + d_{setup}(v)}{1 + l_{s(v)} - l_{s(u)}}$ for any edge (u, v) according to its definition. Therefore, T_{max} is a feasible cycle period strictly smaller than T_{opt} , which contradicts the assumption that T_{opt} is the optimal cycle period. ■

The implication of Lemma 1 is that T_{opt} is determined by $D(u, v)$, $l_{s(u)}$ and $l_{s(v)}$ on some edge (u, v) , which helps us quickly arrive at the following theorem.

Theorem 1: T_{opt} must be in a finite set of size $k(k-1)|E|$. The prescribed-domain clock scheduling can be solved in $O(k^2|V||E|^2)$ time.

Proof: According to Lemma 1, $T_{opt} \in \{\frac{D(u, v) + d_{setup}(v)}{1 + l_{s(v)} - l_{s(u)}} \mid (u, v) \in E, s(u), s(v) \in \{1, \dots, k\}\}$. There are at most $k(k-1)$ distinct values for all differences of $l_{s(v)}$ and $l_{s(u)}$. In addition, there are at most $|E|$ distinct

values for all $D(u, v)$'s. Therefore the possible values of T_{opt} can be constrained into a finite set with size of $k(k-1)|E|$. For a candidate of optimal cycle period, the feasibility check can be finished in $O(|V||E|)$ time by running a modified Bellman-Ford algorithm [19] in G' . In summary, T_{opt} can be found in $O(k^2|V||E|^2)$ time. ■

Theorem 1 reveals an important fact. Although the problem of prescribed-domain clock skew scheduling is not monotone, the number of its breaking point in its solution space is polynomial. This special structure allows us to design an efficient algorithm to solve this problem. First, we estimate a lower bound T_l and an upper bound T_u for T_{opt} such that $T_l \leq T_{opt} \leq T_u$. We set T_l as the optimal cycle period without hold constraints, which can be obtained using a fast incremental algorithm in [7]. T_u is set to $\max_{(u, v) \in E} D(u, v)$, which is the longest combinational path delays among all paths. Then the candidates set of possible T_{opt} is computed by enumerating all possible combinations as shown in Theorem 1. Notice that not all candidates satisfy the condition $T_l \leq T \leq T_u$, we discard those out of bound directly and get the final candidate set S_T , which is sorted in ascending order. Next, we check the feasibility of candidate T in S_T sequentially starting from T_l . The algorithm terminates when we find the first feasible T , which is the optimal cycle period. Given T_{opt} , a feasible clock skew l of each flip-flop can be gained by the potential of the corresponding node in the end of Bellman-Ford algorithm. The general flow of our algorithm is as shown in Algorithm 1.

Algorithm 1 Optimal Algorithm for PDCSS Problem

- 1: Estimate the lower bound and upper bound of the optimal clock period T_{opt} that $T_l \leq T_{opt} \leq T_u$;
 - 2: Construct the candidate optimal clock period set $S_T = \{T = \frac{D(u, v) + d_{setup}(v)}{1 + l_{s(v)} - l_{s(u)}} \mid T_l \leq T \leq T_u, (u, v) \in E, s(u), s(v) \in \{1, \dots, k\}\}$;
 - 3: Sort S_T in ascending order;
 - 4: `currentIndex` = 0, $T = S_T[\text{currentIndex}]$;
 - 5: **while** $T_l \leq T \leq T_u$ **do**
 - 6: **if** T is feasible **then**
 - 7: return T ;
 - 8: **else**
 - 9: `currentIndex++`, $T = S_T[\text{currentIndex}]$;
 - 10: **end if**
 - 11: **end while**
-

It is worth mention the relationship between time complexity of different PDCSS and MDCSS algorithms and domain number k . The proposed PDCSS algorithm has $O(k^2)$ run-time while the first step of the method proposed in [7] has $O(k)$ run-time. However, it is the second step of delay-padding in [7] that dominates the overall performance for PDCSS. As for optimal MDCSS algorithm it has been proved that the time needed to get the exact solution can augment exponentially with k , due to the hardness of the problem itself [17]. The runtime for large circuits of algorithm proposed in [10] could be acceptable only when $k \leq 4$. Though [10] claimed that most circuits can be optimized within a few number of

domains, [15] has shown that there does exist some design circumstances that require a moderate number of domains. On this aspect, the proposed PDCSS algorithm has its advantage over MDCSS when dealing with large and complex circuits.

IV. EXPERIMENTS

A. Experimental Settings

We implemented our algorithm for PDCSS in C++ and tested it on a Linux server with 2.0GHz Intel processor and 2GB memory. ISCAS89 benchmarks are used to evaluate the algorithm and its performance on real circuits. The ISCAS benchmarks were technology mapped through SIS using the library *lib2.genlib*, which is the same as in [10]. Since the other existing method for PDCSS proposed in [7] spends most of the runtime on delay-padding, and doesn't consider the hold constraints, we omit the comparison with [7]. Instead, we will compare the performance and quality of our result with the most advanced MDCSS algorithms.

To demonstrate the effectiveness of PDCSS, we use the same prescribed domain skews for all benchmarks in test with certain domain number. We observed that the optimal clock phase shifts are rarely uniform-distributed in real circuits, instead, they tend to concentrate on small skews rather than large skews. Thus in order to properly handle both small clock skews and large clock skews, we use a "expanded" mode to generate prescribed domains, i.e. the distribution of clock domains are relatively dense on small skews and relatively sparse on large skews. Our experiments show that on average the expanded-distributed domains serve different circuits better than uniform-distributed domains. We use two sets of different prescribed-domains in our test. For $K = 4$ domains, the possible domains are $\{0, \frac{1}{16}T, \frac{3}{16}T, \frac{7}{16}T\}$. For $K = 6$ domains, the possible domains are $\{0, \frac{1}{16}T, \frac{2}{16}T, \frac{4}{16}T, \frac{6}{16}T, \frac{9}{16}T\}$.

B. Speed-up Technology

In the implementation, we employ various heuristics to speed up the exhaustive search in our algorithm. First, we have a tight lower bound T_l and upper bound T_u for optimal cycle period T_{opt} . All the candidates produced by Equation (4) that are out of bound (invalid) would not be taken into account. According to our experiments, on average only 10% candidates in the total $k(k-1)|E|$ will be within the bound. The proportion of valid candidates will even further decrease since most paths in large circuits are uncritical. In addition, we start searching from T_l and stop when the first feasible T is found, thus not all valid candidates in S_T will be checked. Actually, in experiments we find that only a small portion of candidates in S_T will go into the stage of feasibility check since T_{opt} is usually close to T_l . Therefore the total number of candidate T that will be checked is much less than the theoretical bound $k(k-1)|E|$. Statistics show that the number of checked candidates is typically less than 0.5% of the bound $k(k-1)|E|$. Moreover, we utilize a modified Bellman-Ford algorithm to check the feasibility of a candidate T , which is in essence the same as negative-cycle detection [20]. In negative-cycle detection, each edge in E will be relaxed in each

iteration, if we find that some node potential is still decreasing after $|V|$ iterations, then a negative cycle is detected. The total time complexity of negative cycle detection is $|V||E|$. In our feasibility check, each node (flip-flop) has only k possible clock skews. Thus the total number of iterations will be bounded by kL , where L is the length of longest cycle in the circuit. This is because that the "negative signal" can propagate through the a negative cycle for at most k times before some node on the cycle reaches the skew limit. Another heuristic to speed up feasibility check is to mark the relaxed node in each iteration. Only those nodes that are updated in last iteration will be considered for relaxation in the current iteration. We will terminate the iteration as long as there is no new relaxation or some node hits its skew limit. These heuristics turn out to be very helpful in practice, which render a much less runtime than the theoretical bound $O(k^2|V||E|^2)$.

C. Results Comparison

The results are collected in Table I. Column "#V" and "#E" give the number of vertices and edges in the constraint graph. Column " T_{opt} " gives the optimal cycle periods in unconstrained clock skew scheduling, which is also a lower bound of the optimal clock periods in MDCSS. For convenience of comparison, all cycle periods are normalized against T_{opt} . The optimal cycle period ratio, actual domain number used and running time are shown in Column " T/T_{opt} ", "Actual #Domain" and "Runtime", respectively. The results of cycle period for 2,3,4 domains in [10], [12] are also listed as a reference. Some results in [12] is not available since not reported.

Our experiments show competitive cycle periods compared to MDCSS with the same domain number for most cases. Notice that in a small number of cases that the gap between our solution and optimal solution is notable such as *s344*, which is probably caused by the very small skews (smaller than minimum interval between prescribed domains) needed in the optimization. Although our solution is slightly worse than optimal MDCSS as expected, the proposed algorithm shows much better scalability with domain number k . Even for the worst case it only takes less than 2 seconds for 6 domains. In contrast, a domain number more than 4 is computationally prohibited for approach in [10]. To the best of our knowledge, there is no existing work for MDCSS that has provable scalability with domain number k . The proposed algorithm finishes most of the cases instantly (<0.01 sec) and takes up to 0.4 second for worst case when there are 4 domains. In [10], the authors didn't report the exact runtime, they claimed that the last 4 large circuits would take minutes for 4 domains. Our algorithm is also faster than the heuristics proposed in [12]. They only reported the runtime for 3 large benchmark *s35932*, *s38584*, *s38417* as 1, 1, 2 second. As for our algorithm the runtime is 0.01, 0.07, 0.01 second, respectively.

V. CONCLUSION

In this paper, we proposed the first efficient and optimal algorithm for prescribed-domain clock skew scheduling which handles the setup and hold constraints simultaneously. Without

TABLE I
RESULTS OF PRESCRIBED-DOMAIN CLOCK SKEW SCHEDULING ON ISCAS89 SEQUENTIAL BENCHMARKS

Design	#V	#E	T_{opt}	Proposed						Referred T/T_{opt}					
				T/T_{opt}		Actual #Domains		Runtime (s)		SAT-based[10]			M-clustering[12]		
				$K=4$	$K=6$	$K=4$	$K=6$	$K=4$	$K=6$	$k=2$	$k=3$	$k=4$	$k=2$	$k=3$	$k=4$
s27	5	21	5.06	1.03	1.03	3	4	<0.01	<0.01	1.14	1.00	1.00	-	-	-
s208	10	70	9.91	1.00	1.00	3	3	<0.01	<0.01	1.00	1.00	1.00	-	-	-
s298	16	86	10.79	1.03	1.03	3	3	<0.01	<0.01	1.05	1.00	1.00	1.07	1.07	1.05
s344	17	121	13.15	1.09	1.05	3	3	<0.01	<0.01	1.09	1.00	1.00	1.09	1.00	1.00
s349	17	121	13.51	1.09	1.05	3	3	<0.01	<0.01	1.09	1.01	1.00	1.09	1.00	1.00
s382	23	175	9.63	1.07	1.02	3	4	<0.01	<0.01	1.20	1.01	1.00	1.20	1.03	1.03
s386	8	129	9.61	1.04	1.04	2	3	<0.01	<0.01	1.04	1.00	1.00	1.04	1.00	1.00
s400	23	175	9.89	1.08	1.02	3	5	<0.01	<0.01	1.17	1.03	1.02	1.18	1.03	1.03
s420	18	146	21.13	1.00	1.00	2	2	<0.01	<0.01	1.00	1.00	1.00	-	-	-
s444	23	175	8.10	1.20	1.14	4	6	<0.01	<0.01	1.34	1.18	1.10	1.39	1.18	1.15
s510	8	103	14.29	1.00	1.00	1	1	<0.01	<0.01	1.00	1.00	1.00	1.00	1.00	1.00
s526	23	167	11.22	1.02	1.02	3	3	<0.01	<0.01	1.05	1.00	1.00	1.07	1.07	1.05
s526n	23	167	11.31	1.02	1.02	3	2	<0.01	<0.01	1.05	1.00	1.00	-	-	-
s641	21	486	29.51	1.00	1.00	2	2	<0.01	<0.01	1.00	1.00	1.00	1.00	1.00	1.00
s713	21	486	30.58	1.00	1.00	1	1	<0.01	<0.01	1.00	1.00	1.00	1.00	1.00	1.00
s820	7	213	16.74	1.00	1.00	1	1	<0.01	<0.01	1.00	1.00	1.00	-	-	-
s832	7	213	16.22	1.00	1.00	1	1	<0.01	<0.01	1.00	1.00	1.00	1.00	1.00	1.00
s838	34	298	44.66	1.00	1.00	2	2	<0.01	<0.01	1.00	1.00	1.00	-	-	-
s953	8	94	14.38	1.06	1.06	3	2	<0.01	<0.01	1.03	1.00	1.00	-	-	-
s1196	19	365	22.28	1.00	1.00	1	1	<0.01	<0.01	1.00	1.00	1.00	1.00	1.00	1.00
s1238	19	365	24.33	1.03	1.00	2	3	<0.01	<0.01	1.00	1.00	1.00	1.03	1.03	1.01
s1423	76	2235	73.13	1.04	1.02	2	2	<0.01	0.01	1.03	1.01	1.00	1.05	1.04	1.04
s1488	8	266	23.18	1.00	1.00	2	2	<0.01	<0.01	1.00	1.00	1.00	1.00	1.00	1.00
s1494	8	266	23.85	1.00	1.00	2	2	<0.01	<0.01	1.00	1.00	1.00	1.00	1.00	1.00
s5378	165	2180	22.88	1.19	1.12	2	3	<0.01	0.01	1.13	1.01	1.00	-	-	-
s9234	140	2226	33.76	1.00	1.00	2	2	<0.01	<0.01	1.01	1.00	1.00	-	-	-
s13207	471	3885	53.36	1.03	1.03	2	2	<0.01	<0.01	1.03	1.00	1.00	-	-	-
s15850	565	16375	85.27	1.09	1.09	2	2	0.39	1.51	1.08	1.04	1.04	-	-	-
s35932	1442	6128	286.24	1.01	1.01	1	1	<0.01	<0.01	1.00	1.00	1.00	1.01	1.01	1.01
s38417	1465	31980	86.19	1.02	1.02	1	1	0.07	0.12	1.00	1.00	1.00	1.00	1.00	1.00
s38584	1451	17900	286.62	1.00	1.00	1	1	0.01	0.01	1.00	1.00	1.00	1.00	1.00	1.00

additional cost to fix hold constraints, our algorithm improves the previous work on prescribed-domain clock skew scheduling and yields competitive solution quality while running in much less time with respect to multi-domain clock skew scheduling. The fruit of our work may promote the use of the designer-friendly prescribed-domain clock skew scheduling in real applications.

ACKNOWLEDGEMENT

This work is supported by the NSF under CCF-0811270 and CCF-1115550.

REFERENCES

- [1] M. A. B. Jackson, A. Srinivasan, and E. S. Kuh, "Clock routing for high performance ic's," in *IEEE Proc. DAC*, 1990.
- [2] A. Kahng, J. Cong, and G. Robins, "High performance clock routing based on recursive geometric matching," in *IEEE Proc. DAC*, 1991.
- [3] S. Pullela, N. Menezes, and L. T. Pillage, "Reliable nonzero clock skew trees using wire width optimization," in *IEEE Proc. DAC*, 1993.
- [4] —, "Skew and delay optimization for reliable buffered clock trees," in *IEEE Proc. ICCAD*, 1993.
- [5] —, "Exact zero skew," in *IEEE Proc. ICCAD*, 1991, pp. 336–339.
- [6] J. Fishburn, "Clock skew optimization," *IEEE Trans. on Comput.*, vol. 39, no. 7, pp. 945–951, 1990.
- [7] C. Lin and H. Zhou, "Clock skew scheduling with delay padding for prescribed skew domains," in *IEEE Proc. ASPDAC*, 2007, pp. 541–546.
- [8] S. Huang, C. Chuang, and Y. Nieh, "Fast multi-domain clock skew scheduling for peak current reduction," in *IEEE Proc. ASPDAC*, 2006, pp. 254–259.
- [9] A. Vittal, H. Ha, F. Brewer, and M. Marek-Sadowska, "Clock skew optimization for ground bounce control," in *IEEE Proc. ASPDAC*, 2007, pp. 395–399.
- [10] K. Ravindran, A. Kuehlmann, and E. Sentovich, "Multi-domain clock skew scheduling," in *IEEE Proc. ICCAD*, 2003, pp. 801–808.
- [11] N. MohammadZadeh, M. Mirsaedi, A. Jahanian, and M. S. Zamani, "Multi-domain clock skew scheduling-aware register placement to optimize clock distribution network," in *IEEE Proc. DATE*, April 2009, pp. 833–838.
- [12] J. Casanova and J. Cortadella, "Multi-level clustering for clock skew optimization," in *IEEE Proc. ICCAD*, 2009, pp. 547–554.
- [13] Y. Zhi, H. Zhou, and X. Zeng, "A practical method for multi-domain clock skew optimization," in *IEEE Proc. ASPDAC*, 2011, pp. 521–526.
- [14] Y. Zhi, W.-S. Luk, H. Zhou, C. Yan, H. Zhu, and X. Zeng, "An efficient algorithm for multi-domain clock skew scheduling," in *IEEE Proc. DATE*, 2011.
- [15] M. Ni and S. O. Memik, "A fast heuristic algorithm for multidomain clock skew scheduling," *IEEE Trans. on VLSI*, vol. 18, no. 4, pp. 630–637, 2010.
- [16] N. MohammadZadeh, M. Mirsaedi, A. Jahanian, and M. S. Zamani, "Multi-domain clock skew scheduling-aware register placement to optimize clock distribution network," in *IEEE Proc. DATE*, April 2009, pp. 833–838.
- [17] L. Li, Y. Lu, and H. Zhou, "Optimal multi-domain clock skew scheduling," in *IEEE Proc. DAC*, 2011.
- [18] N. V. Shenoy, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "Minimum padding to satisfy short path constraints," in *IEEE Proc. ICCAD*, 1993.
- [19] J. P. Fishburn, "Solving a system of difference constraints with variables restricted to a finite set," *Inf. Process. Lett.*, vol. 82, no. 3, pp. 143–144, 2002.
- [20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms (Second Edition)*. The MIT press, 2001.