Potential Slack Budgeting with Clock Skew Optimization

Kai Wang and Malgorzata Marek-Sadowska

Department of Electrical and Computer Engineering
University of California, Santa Barbara, CA 93106-9560, USA
{wk,mms}@ece.ucsb.edu

Abstract

Potential slack is an effective metric of circuit's possible performance improvement. It is equal to the maximal amount of slack that can be potentially used for optimization. In this paper, we first present a new, linear programming-based approach for potential slack calculation. Our method produces an optimal solution with significant runtime speedup compared to previous methods. Then we formulate and solve the problem of global potential slack budgeting by clock-skew optimization. We demonstrate experimentally that the potential slack can be significantly improved by appropriate clock skew assignment.

1. Introduction

The performance of integrated circuits is measured by three major design objectives: timing, area, and power dissipation. The goal of performance optimization is either to minimize the circuit area and power dissipation under timing constraints, or to minimize the critical path delay for a given area/power budget. A design process consists of many intermediate design stages. At different stages, various optimization strategies are adopted to improve the overall system performance.

In [1] the authors introduce two types of performance: *immediate* and *potential*. The immediate performance can be obtained using existing estimation techniques. In contrast the potential performance indicates how much improvement could be made through future optimization. Because design decisions made at higher levels have great impact on the optimization results at the lower levels, by estimating potential performance, especially during the early stages of design, designers can avoid the tedium of going through the optimization processes at lower levels. Combined with immediate performance estimation, potential performance prediction enables efficient exploration of design space [1].

A slack of a module in a combinational circuit is the difference between its arrival time and required time. Most of the prior works on slack management or delay budgeting ([4], [5], [6], [9]) focus on assigning values to the delay slacks (thus assigning values to the signal arrival and required times) of the modules. The optimization objective is usually to maximize some increasing cost function of slacks, or to generate delay constraints for timing-driven placement and routing. These techniques work with the immediate performance, because they consider only the existing slack in the circuit.

For timing-constrained problems, potential performance improvement in terms of area or power is usually obtained from the slacks of the noncritical parts in the circuit. This is because for noncritical parts, the delay penalty caused by area/power improvement will not necessarily degrade the circuit timing. However, not all of the total existing slack can be utilized for optimization because the slacks of different modules affect each other depending on the topological structure of circuits. The concept of *potential slack* was first introduced in [1] to measure a circuit's potential performance based on slack. Its applications on gate sizing and timing-driven placement were also discussed. When the potential slack of a circuit implementation is known, the designer can predict the potential area/power reduction without having to go through actual low-level area/power optimization. For example, the authors of [1] conducted gate-sizing experiments after calculating potential slack. Their results showed that for all the tested circuits, the best implementations predicted by potential slack led to maximum area reduction. While the potential slack provides 100% correct prediction, the other commonly used metrics are not so well correlated with performance. The longest path delay has a 20% chance of giving a correct prediction, and total slack has a 40% chance of predicting the achievable performance results. Therefore, we can predict or maximize the possible area/power reduction by dealing only with potential slack without going through gate sizing, whose computation cost is much higher.

In [1], two algorithms are proposed to find potential slacks of given circuits. One is based on the maximum-independent-set algorithm (*MISA*), which can provide optimal solutions but its computation cost is high; the other is a greedy estimation algorithm, which is much faster than *MISA* but generates sub-optimal solutions. Besides, these methods focus only on computing potential slack of a single combinational logic block. However, potential slack of a logic block is affected by the signal arrival and required times at the primary inputs and outputs, which in turn are decided by the arrival times of the clock signals at the launching flip-flops. The clock arrival times can be adjusted through clock skew optimization. For a system consisting of a set of combinational logic blocks and a clock distribution network feeding clock signal to each block, a synergistic blend of potential slack calculation and clock skew optimization could improve the total potential slack of all logic blocks in the system.

In this paper, we first propose a new approach for potential slack calculation based on linear programming. Our approach determines optimal solutions with significant run-time speedup in comparison to the existing methods. Then we combine our technique with clock skew optimization to maximize the total potential slack of all logic blocks subject to clock-skew constraints.

The rest of the paper is organized as follows. Preliminaries are given in section 2. We describe our linear-programming-based approach in section 3. In section 4, we discuss how to combine the proposed technique with clock-skew optimization for global potential slack budgeting. In section 5 we give the experimental results. We conclude the paper in section 6.

2. Preliminaries

In this section we briefly review the concept of potential slack and its calculation methods. We follow the definitions and algorithms used in [1].

Consider a combinational circuit composed of n modules $V = \{v_1, v_2, ..., v_n\}$, and m nets $E = \{e_1, e_2, ..., e_m\}$. Such a circuit can be modeled as a *directed acyclic graph* (*DAG*) *G*. In the graph, each node represents a module, and there is an edge from node v_i to node v_j if and only if the output of v_i is an input of v_j. Each node v_i ($i = 1, ..., n$) is associated with a propagation delay d_i. We assume that the arrival times for all primary inputs are zero and the required times for all primary outputs are the specified timing requirements. For each node v_i, the arrival time a_i and required time r_i can be computed recursively as follows:

$$\begin{aligned} a_i &= max(a_j + d_i + \delta_{ji}) \qquad v_j \in FI(v_i) \\ r_i &= min(r_k - d_k - \delta_{ik}) \qquad v_k \in FO(v_i) \end{aligned} \tag{1}$$

where $FI(v_i)$ ($FO(v_i)$) is a set of fanin (fanout) nodes of a node v_i. δ_{ji} is the interconnect delay from a node v_j to a node v_i. To simplify our discussion, from now on we assume that all interconnect delays are zero. Our technique can be easily extended to consider interconnect delays.

The slack of a node v_i is defined as:

$$s_i = r_i - a_i \tag{2}$$

The vector of slacks $\overrightarrow{S(V)} = [s_1, s_2, ..., s_n]$ is called the *slack distribution* of the circuit, and the total slack is:

$$\left|\overrightarrow{S(V)}\right| = \sum_{i=1}^{n} s_i \quad . \tag{3}$$

The circuit is said to be *safe* if and only if the slack of every module is non-negative, *i.e.*,

$$\overrightarrow{S(V)} \geq 0$$

If we increase the delays of some modules, it will cause a change of the circuit's slack distribution. A *slack assignment* is defined as a non-negative vector of incremental delays, which updates the slack distribution from $\overrightarrow{S(V)}$ to $\overrightarrow{S_\Delta(V)}$:

$$\overrightarrow{\Delta D(V)} = [\Delta d_1, \Delta d_2, \ldots, \Delta d_n] \geq 0 \tag{4}$$

If $\overrightarrow{S_\Delta(V)}$ is non-negative, $\overrightarrow{\Delta D(V)}$ is said to be an *effective assignment* because the increased delay for each module can be effectively transferred to area/power reduction in future optimization. The *effective slack* for this assignment is defined as:

$$\left|\overrightarrow{\Delta D(V)}\right| = \sum_{i=1}^{n} \Delta d_i \tag{5}$$

The maximum of effective slack is called *potential slack* (*PS*) of the circuit. A slack assignment is *optimal* if it leads to the potential slack. The following two lemmas were proved in [1]:

Lemma 1: If a slack assignment leads to the potential slack, then the resulting slack distribution is $\overrightarrow{S_\Delta(V)} = 0$.

Lemma 2: For any safe circuit, the total slack is an upper bound of potential slack.

Fig. 1 shows a simple example, where the arrival times for the inputs are all zero, the required time for the output (node v_5) is 6, and the initial delay of each node is 1. The initial slack distribution is [3,3,3,3,3], and the total slack is 15. If we apply a slack assignment [1,1,0,0,0] to the circuit, the slack distribution becomes [2,2,2,2,2]. The effective slack assignment [0,0,3,3,0] leads to the potential slack $PS = 6$ and the resulting slack distribution is zero. The slack assignment of [1,1,1,1,1] also leads to zero slack distribution, but it is not the optimal assignment because its effective slack is $5 < PS = 6$.

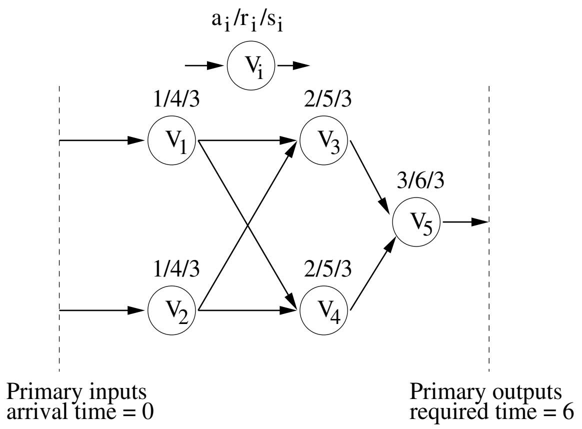

Figure 1: Slack distribution

While finding total slack is straightforward, calculating the potential slack is not trivial. In [1], an algorithm based on maximum independent set (*MISA*) was proposed to solve the problem of potential slack calculation. This algorithm works as follows: during each iteration, a slack-equalization graph G_m is constructed. G_m contains the nodes with maximal slacks in the original graph *G*. There is an edge between two nodes in G_m if these nodes are slack sensitive to each other (change of slack of one node affects the slack of the other node) in *G*. G_t is a transitive closure of the slack-equalization graph G_m. In G_t a maximum independent set (*MIS*) of nodes is selected. To each node in *MIS*, an incremental delay of $S_m - S_{m-1}$ is assigned, where S_m and S_{m-1} are the maximum and the second largest slack. This procedure continues till the slack distribution becomes zero. Although *MISA* can produce the optimal solution, its time complexity is $o(Kn^3)$, where *n* is the number of nodes, and *K* is the number of different slacks in the original graph. Consequently, the computation cost of *MISA* is of a major concern. To estimate quickly the potential slack, a greedy algorithm is also described in [1]. It selects nodes for which to assign additional delay based on the local slack information. This greedy algorithm is much faster than *MISA*, but it cannot produce optimal solutions due to its lack of global view of the circuit structure.

3. Linear Programming based Approach

Initially, we are given a circuit and specified arrival times at primary inputs, and required times at primary outputs. We attach a zero-delay pseudo-module to each primary input and primary output, and then we model the circuit as a *DAG G* as described in section 2. The linear program is formulated as follows:

Objective function:

$$Maximize \qquad \sum_{i=1}^{n} \Delta d_i \tag{6}$$

where *n* is the number of nodes in the graph, and Δd_i is the incremental delay of a node v_i.

Constraints:

1) Arrival/required time constraint

for a node $v_i \in PI$:

$$a_i = ART_i \tag{7}$$

for a node $v_i \in PO$:

$$r_i \le RET_i \tag{8}$$

for a node $v_i \notin PI \cup PO$:

$$\begin{matrix} a_i \ge a_j + d_i^0 + \Delta d_i \\ v_j \in FI(v_i) \qquad i = 1 \ldots n \end{matrix} \tag{9}$$

$$\begin{matrix} r_i \le r_k - (d_k^0 + \Delta d_k) \\ v_k \in FO(v_i) \qquad i = 1 \ldots n \end{matrix} \tag{10}$$

where d_i^0 is the initial delay of a node v_i, *PI (PO)* are the primary input (output) nodes of the graph, and ART_i (RET_i) is the specified arrival (required) time.

2) Zero slack distribution constraint

Lemma 1 in section 2 states that the slack distribution resulting from the optimal slack assignment is zero, because for any safe circuit, if $s_i > 0$, we can always assign to a node v_i an additional delay $\Delta d_i = s_i$ such that $s_i = 0$ after the assignment. Note that the reverse of Lemma 1 is not necessarily true. Therefore we have the following constraint for the optimal solution of potential slack:

$$a_i = r_i \qquad i = 1 \ldots n \tag{11}$$

We have the following theorem for the optimality of the above linear programming approach.

Theorem 1: The optimal solution for the above linear programing problem gives the optimal slack assignment.

Proof: We only need to prove that for the optimal solution of the linear program, the constraints in Eqn.(1) hold. We first consider the case of arrival time. For each node v_i, at least one of the equality in Eqn.(9) holds. Because if not, for every $v_j \in FI(v_i)$, we have

$$a_i > a_j + d_i^0 + \Delta d_i$$

and we can assign an additional delay $\Delta d'_i$ to a node v_i such that the value of the objective function is increased by:

$$\Delta d'_i = min(a_i - a_j - d_i^0 - \Delta d_i) \qquad v_j \in FI(v_i)$$

However, this contradicts the assumption that the obtained solution is the optimal one for the linear program. Similarly, we can prove the case of required time. Therefore, under the optimal solution of the linear program, the definitions of arrival and required times in Eqn.(1) hold, which means that the optimal slack assignment is achieved.

Therefore for each node v_i, we can use one variable τ_i to represent both the arrival and required times, *i.e.*

$$\tau_i = a_i = r_i \tag{12}$$

Thus we can remove the zero slack distribution constraint in Eqn.(11), and reduce the number of variables of the linear program.

Fig. 2 shows an example illustrating the above formulation. The arrival times for primary inputs are all 0, and the required times for the primary outputs are 10. The initial delays are all 1.

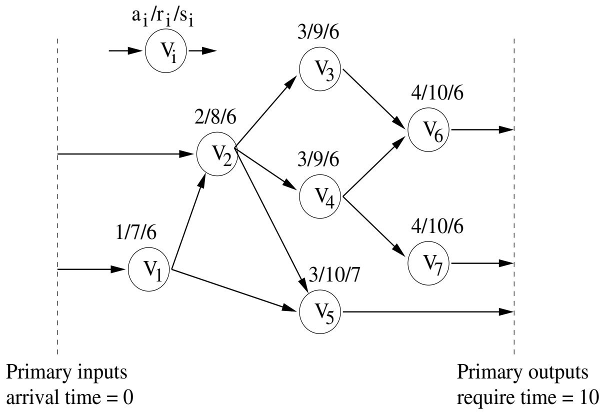

Figure 2: Example of LP formulation

We can formulate the linear program as follows:

$$Maximize \qquad \sum_{i=1}^{7} \Delta d_i$$

subject to:

$$\tau_1 \geq 1 + \Delta d_1,\ \tau_2 \geq 1 + \Delta d_2$$
$$\tau_2 \geq \tau_1 + 1 + \Delta d_2,\ \tau_3 \geq \tau_2 + 1 + \Delta d_3$$
$$\tau_4 \geq \tau_2 + 1 + \Delta d_4,\ \tau_5 \geq \tau_2 + 1 + \Delta d_5$$
$$\tau_5 \geq \tau_1 + 1 + \Delta d_5,\ \tau_6 \geq \tau_3 + 1 + \Delta d_6$$
$$\tau_6 \geq \tau_4 + 1 + \Delta d_6,\ \tau_7 \geq \tau_4 + 1 + \Delta d_7$$
$$\tau_5 \leq 10,\ \tau_6 \leq 10,\ \tau_7 \leq 10$$

Solving this linear programming problem gives us the potential slack PS = 19, and the optimal slack assignment is [0,0,6,6,7,0,0].

4. Global Budgeting for Potential Slack with Clock Skew Optimization

Clock skew optimization is a well-known technique for determining intentional skews of clock sinks to improve the system performance or reliability. Consider a synchronous circuit with positive edge-triggered flip-flops in a single-phase clocking scheme. Let FF_i and FF_j be two sequentially adjacent flip-flops, with FF_i feeding data to FF_j, and a combinational logic block between them. The signal arrival time at the clock pin of FF_i (FF_j) is c_i (c_j). To ensure correct logic operations, we must bound the skew between FF_i and FF_j from above and below by the following constraints:

$$\begin{gathered} c_i - c_j \geq t_{hold} - d_{pFF} - d_{\text{logic}}^{min} \\ c_i - c_j \leq C_P - d_{pFF} - d_{\text{logic}}^{max} - t_{setup} \end{gathered} \tag{13}$$

where d_{logic}^{min} and d_{logic}^{max} are the minimum and maximum delays through the combinational logic; d_{pFF} is the delay through the flip-flop; t_{hold} and t_{setup} are the hold and setup times for flip-flop; and C_p is the clock cycle.

The clock arrival times of the launching flip-flops at the boundary of a logic block affect the data arrival and required times at the block's primary inputs and outputs, which in turn decide the potential slack of this block. The clock arrival times can be adjusted through clock skew optimization. Therefore, clock skew optimization could be used to improve the total potential slack of all the logic blocks in the system.

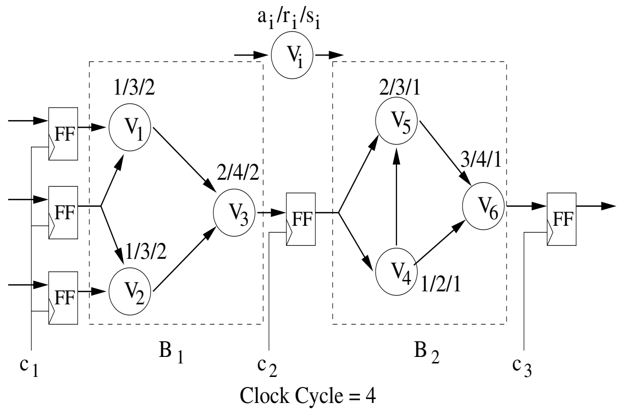

Figure 3: Potential slack budgeting with clock skew optimization

In Fig. 3 we show a small system composed of two logic blocks B_1 and B_2, with flip-flops at their inputs and outputs. For simplicity, we assume that the delay, hold time, and setup time for the flip-flops are zero. The clock cycle is 4, and the clock arrival times are c_1, c_2, and c_3. The initial delay of each module is 1. With a zero-skew schedule, we have $c_1 = c_2 = c_3 = 0$. The potential slack of B_1 is $PS(B_1) = 4$, the potential slack of B_2 is $PS(B_2) = 1$; thus the total potential slack of the system is 5. However, if we change c_2 to 1, and c_3 to 2, we have $PS(B_1) = 6$, $PS(B_2) = 2$, and the total potential slack becomes 8.

The problem of global potential slack budgeting with clock-skew optimization is to decide a clock skew schedule such that the total potential slack of all logic blocks is maximized. Due to the linear nature of the proposed technique for potential slack calculation, we can combine it with the clock skew constraints to form a unified linear programming problem as follows:

Objective function:

$$maximize \qquad \sum_{j=1}^{N} \sum_{i=1}^{n_j} \Delta d_{ij} \tag{14}$$

where N is the number of combinational logic blocks, n_j is the number of nodes in the jth block, and Δd_{ij} is the incremental delay variable for the ith node in the jth block.

Constraints:

for each block B_j ($j = 1,..., N$) and each node v_{ij} ($i = 1,..., n_j$) inside B_j, we have

for $v_{ij} \in PI_j$

$$\tau_{ij} = c^{PI}{}_j + d_{pFF} + t_{hold} \tag{15}$$

for $v_{ij} \in PO_j$

$$\tau_{ij} \le c^{PO}{}_j + C_p - t_{setup} \tag{16}$$

$$\tau_{ij} \ge c^{PO}{}_j - c^{PI}{}_j + t_{hold} - d_{pFF} \tag{17}$$

for $v_{ij} \notin PI_j \cup PO_j$:

$$\tau_{ij} \ge \tau_{kj} + d_{ij}^{\ 0} + \Delta d_{ij} \qquad v_{kj} \in FI(v_{ij}) \tag{18}$$

$$\tau_{ij} \le \tau_{kj} - (d_{kj}^{\ 0} + \Delta d_{kj}) \qquad v_{kj} \in FO(v_{ij}) \tag{19}$$

where τ_{ij} is the variable representing both the arrival time and required time of a node v_{ij}. $c^{PI}{}_j$ is the clock arrival time at the primary input flip-flops of the block B_j, and $c^{PO}{}_j$ is the clock arrival time at the primary output flip-flops of the block B_j.

In [2], clock skew optimization and gate sizing are combined to minimize the sequential circuit area. However, this technique is not suitable for large circuits because gate sizing for all logic blocks simultaneously is computationally infeasible. In contrast, our approach maximizes the potential slack over all the blocks before going through the low-level gate sizing step. The strong correlation between potential slack and actual area reduction will lead to desired area reduction after the actual gate sizing.

5. Experimental Results

We have developed our prototype tool using C++ programming language based on the proposed technique. The linear programs are solved using the *Xpress* optimization package [8]. For comparison, the *MISA* and greedy algorithms described in [1] are also carefully implemented. All experiments are carried out on a *P4* 2.4GHz PC running Linux.

5.1. Potential slack calculation

In the first experiment, we compare our linear-programming-based potential slack calculation approach with *MISA* and the greedy algorithm [1]. We test each algorithm on a set of *MCNC* benchmarks, which are implemented and mapped into a $0.13\mu m$ technology library using *SIS* [7].

The results are summarized in Table 1. Columns 1 through 3 give the circuit name, the number of gates, and the total slacks. Column 4 lists the potential slacks obtained using *MISA*, column 5 shows the potential slack obtained from the greedy algorithm, and column 6 shows the results of our linear programming approach. Column 7 gives the percentage improvement of potential slack computed with *LP* or *MISA* compared to the greedy approach. Columns 8 through 10 give the *CPU* times in seconds for different approaches.

From Table 1, we observe that for all benchmarks, the potential slacks produced by *MISA* and by our linear programming approach are the same, which validates empirically the optimality of the linear programming approach. The potential slacks produced by greedy algorithm are much smaller than those in the optimal solution.

On the other hand, as shown in Fig. 4, the CPU times used by the linear programming approach are much less than the *MISA* times. For small-size benchmarks (fewer than 3000 gates), the runtimes of the greedy algorithm and of the linear programming approach are comparable, but for large size circuits, our linear programming approach is much faster than the greedy algorithm. These experimental results demonstrate that our linear programming

Table 1: Comparison of MISA, GREEDY and LP

Circuit	Number of Gates	Total Slack	Potential Slack				CPU Time (second)		
			MISA	GREEDY	LP	Improve (%)	MISA	GREEDY	LP
apex7	184	167294	74132	60247	74132	23.0	0.12	0.02	0.01
C432	192	167587	65311	53730	65311	21.6	0.08	0.02	0.04
alu2	283	227729	94902	66307	94902	43.1	0.52	0.01	0.04
C1908	448	515213	113287	76565	113287	47.9	0.81	0.02	0.05
C1355	460	190382	45361	25311	45361	79.2	0.18	0.01	0.04
apex6	510	450577	249662	169604	249662	47.2	0.73	0.04	0.03
dalu	885	2067609	560450	309773	560450	80.9	5.28	0.04	0.1
C5315	1189	1873252	688346	504960	688346	36.3	6.48	0.14	0.12
ex5p	1636	1148174	350928	251733	350928	39.4	14.9	0.25	0.18
C7552	1742	3259573	879996	537157	879996	63.8	22.67	0.15	0.17
i10	1802	4875123	1632618	1356682	1632618	20.3	23.62	0.41	0.19
des	2445	2598367	1197056	752829	1197056	59.0	48.18	0.45	0.25
misex3	2795	1361256	557167	386248	557167	44.3	25.33	1.0	0.36
alu4	2876	1422486	663673	590664	663673	12.3	33.97	1.56	0.41
seq	3393	1837170	792068	669679	792068	18.3	46.81	1.67	0.45
apex2	3891	1866091	751037	658911	751037	13.9	58.28	2.35	0.58
ex1010	4207	2509745	845325	679306	845325	24.4	70.9	2.08	0.63
spla	6906	5096193	1990314	1601929	1990314	24.2	191.36	7.56	1.26
pdc	8857	6158465	2322928	1836508	2322928	26.5	575.16	12.59	1.97

approach is capable of determining potential slacks with significant run-time speedup over the existing methods.

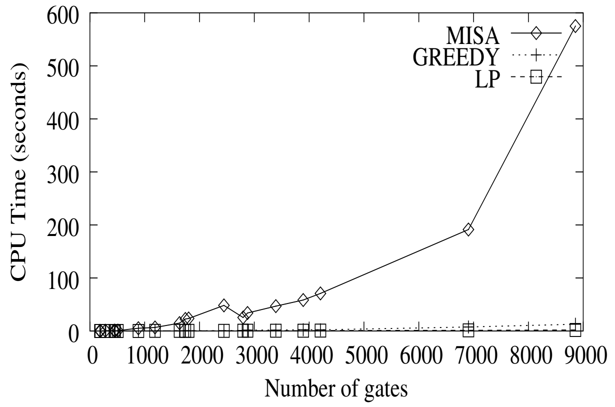

Figure 4: CPU Time vs. Number of gates

5.2. Global budgeting

In the second experiment, we perform global potential slack budgeting with clock-skew optimization on a set of ISCAS'89 benchmarks. For comparison, we implement another approach, which calculates individually the potential slack for each combinational logic block, assuming zero-skew schedule. Finally, to verify the actual area reduction, gate-sizing is carried out for both approaches using the same strategy as in [1]. Specifically, we downsize each node (gate) such that the delay penalty is less than the incremental delay assigned by the potential slack computation algorithm.

The results are summarized in Table 2. Columns 1 through 3 give the circuit name, the number of flip-flops, and the number of gates. Column 4 lists potential slacks obtained using the zero-skew approach, and column 5 lists potential slacks obtained using our global budgeting approach. Column 6 shows the potential slack improvement. Columns 7 and 8 give the actual area reduction percentage for gate sizing following both approaches, and column 9 gives the improvement in area reduction.

We observe that the potential slack determined by the proposed global budgeting technique with clock skew optimization is significantly larger than that produced by the zero-skew approach. Moreover, the larger potential slacks obtained by our global budgeting technique are transformed into larger area reductions, as verified by the actual gate sizing.

Table 2: Global budgeting vs. Zero-skew approach

Circuit	FF. Number	Gate Number	Potential Slack			Area Reduction Percentage after Gate Sizing		
			Zero Skew	Global Budgeting	Improve (%)	Zero Skew	Global Budgeting	Improve (%)
s208	8	104	50231	65278	29.9	19.8%	24.2%	22.2
s1196	18	529	220039	273512	24.3	16.3%	19.6%	20.2
s1423	74	657	284617	332479	16.8	21.1%	23.9%	13.3
s5378	179	2779	1208258	1671921	38.3	25.7%	33.4%	29.9
s13207	669	7951	1889762	2466810	30.5	19.3%	24.6%	27.5
s15850	597	9772	2309835	2938763	27.2	17.9%	22.4%	25.1
s35932	1728	16065	3708499	4480263	20.8	23.2%	27.3%	17.7

6. Conclusions

In this paper we have investigated the problem of potential slack calculation and budgeting. Our main contributions are: 1) we have proposed a linear-programming-based technique for potential slack calculation, which can provide an optimal solution with significant speedup in comparison to the previous approaches; 2) we have combined clock-skew optimization with our technique for potential slack computation to further improve potential slack in the overall system.

Acknowledgment

This work is supported in part by the SRC grant #1069 and in part by the California MICRO program through IBM.

References

[1] C. Chen, X. Yang, and M. Sarrafzadeh, "Predicting potential performance for digital circuits," *IEEE Trans. Computer-Aided Design*, vol. 21, no. 3, Mar. 2002.

[2] W. Chuang, S. S. Sapatnekar, and I. N. Hajj, "A unified algorithm for gate sizing and clock skew optimization to minimize sequential circuit area," *Proc. Intl. Conf. on Computer-Aided Design*, Nov. 1993.

[3] J. P. Fishburn, "Clock skew optimization," *IEEE Trans. Computer-Aided Design*, pp. 945-951, July 1990.

[4] T. Gao, P. M. Vaidya, and C. L. Liu, "A new performance driven placement algorithm," *Proc. Intl. Conf. Computer-Aided Design*, IEEE/ACM, pp. 44-47, 1991.

[5] R. Nair, C. L. Berman, P. S. Hauge, and E. J. Yoffa, "Generation of performance constraints for layout," *IEEE Trans. Computer-Aided Design*, vol. CAD-8, pp. 860-874, Aug. 1989.

[6] M. Sarrafzadeh, D. A. Knol, and G. E. Tellez, "A delay budgeting algorithm ensuring maximum flexibility in placement," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 1332-1341, Nov. 1997.

[7] E. M. Sentovich *et al.*, "SIS: A system for sequential circuit synthesis," Univ. California, Berkeley, UCB/ERL M92/41, 1992.

[8] Xpress^{MP}[Online]: http://www.dashoptimization.com

[9] H. Youssef and E. Shragowitz, "Timing constraints for correct performance," *Proc. Intl. Conf. Computer-Aided Design*, IEEE/ACM, pp. 24-27, 1990.