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:

\[
\begin{align*}
    a_i &= \max(a_j + d_j + \delta_{ji}) \quad &v_j \in FI(v_i) \\
    r_i &= \min(r_k - d_k - \delta_{ik}) \quad &v_k \in FO(v_i)
\end{align*}
\]

where \( FI(v_i) \) (\( FO(v_i) \)) is a set of fanin (fanout) nodes of a node \( v_i \). \( \delta_{ji} \) 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:

\[
s_i = r_i - a_i
\]

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

\[
\mathbf{S}(V) = \sum_{i=1}^{n} s_i
\]

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

\[
\mathbf{S}(V) \geq 0
\]

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 \( \mathbf{S}(V) \) to \( \mathbf{S}(V) \):

\[
\Delta \mathbf{S}(V) = [\Delta d_1, \Delta d_2, ..., \Delta d_n] \geq 0
\]

If \( \mathbf{S}(V) \) is non-negative, \( \Delta \mathbf{S}(V) \) 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:

\[
|\Delta \mathbf{S}(V)| = \sum_{i=1}^{n} \Delta d_i
\]

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 \( \mathbf{S}(V) = 0 \).

**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 \( v_3 \)) is 6, and the initial delay of each node is 1. The initial slack distribution is \([3,3,3,3] \), 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] also leads to zero slack distribution, but it is not the optimal assignment because its effective slack is $5 < PS = 6$.

![Diagram of slack distribution](image)

**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-sensitive 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:**

$$\text{Maximize } \sum_{i=1}^n \Delta d_i$$

where $n$ is the number of nodes in the graph, and $\Delta d_i$ is the incremental delay of a node $v_i$.

**Constraints:**

1) Arrival/required time constraint
   for a node $v_i \in PI$:
   $$a_i = ART_i$$
   (7)

   for a node $v_i \in PO$:
   $$r_i \leq RET_i$$
   (8)

   for a node $v_i \notin PI \cup PO$:
   $$a_i \geq a_j + d_j^0 + \Delta d_i$$
   $$v_j \in FI(v_i) \quad i = 1 \ldots n$$
   $$r_i \leq r_k - (d_k^0 + \Delta d_k)$$
   $$v_k \in FO(v_i) \quad i = 1 \ldots n$$

   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 $s_j > 0$, we can always assign to a node $v_j$ an additional delay $\Delta d_j = s_j$ such that $s_j = 0$ 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:

   $$a_i = r_i \quad i = 1 \ldots n$$
   (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 $v_j$, at least one of the equality in Eqn. (9) holds. Because if not, for every $v_j \in FI(v_i)$, we have

$$a_i > a_j + d_j^0 + \Delta d_i$$
and we can assign an additional delay $\Delta d'_j$ to a node $v_j$ such that the value of the objective function is increased by:

$$\Delta d'_j = \min(a_j - a_i - d'_i - \Delta d_j), \quad v_j \in FI(v_i)$$

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.(11) hold, which means that the optimal slack assignment is achieved.

Therefore for each node $v_j$, we can use one variable $\tau_j$ to represent both the arrival and required times, i.e.

$$\tau_j = a_j = r_j \quad (12)$$

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 $$\sum_{i=1}^n \Delta d_i$$

subject to:

$$\tau_1 \geq 1 + \Delta d_1, \ \tau_2 \geq 1 + \Delta d_2$$
$$\tau_3 \geq \tau_1 + 1 + \Delta d_2, \ \tau_4 \geq \tau_2 + 1 + \Delta d_3$$
$$\tau_5 \geq \tau_2 + 1 + \Delta d_4, \ \tau_6 \geq \tau_3 + 1 + \Delta d_5$$
$$\tau_7 \geq \tau_3 + 1 + \Delta d_5, \ \tau_8 \geq \tau_4 + 1 + \Delta d_6$$
$$\tau_9 \geq \tau_4 + 1 + \Delta d_6, \ \tau_{10} \geq \tau_5 + 1 + \Delta d_7$$
$$\tau_{11} \leq 10, \ \tau_{12} \leq 10, \ \tau_{13} \leq 10$$

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:

$$c_i - c_j \geq t_{\text{hold}} - d_{\text{max}} - C_p - d_{\text{max}} - t_{\text{setup}}$$

$$c_i - c_j \leq t_{\text{hold}} - d_{\text{max}} - C_p - d_{\text{max}} - t_{\text{setup}}$$

where $d_{\text{max}}$ and $d_{\text{max}}$ are the minimum and maximum delays through the combinational logic; $d_{\text{max}}$ is the delay through the flip-flop; $t_{\text{hold}}$ and $t_{\text{setup}}$ 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:**

$$\max \sum_{j=1}^{N_n} \sum_{i=1}^{n_j} \Delta d_{ij}$$  \hspace{1cm} (14)

where $N$ is the number of combinational logic blocks, $n_j$ is the number of nodes in the $j$th block, and $\Delta d_{ij}$ is the incremental delay variable for the $i$th node in the $j$th 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 $v_{ij} \in PI_j$

$$\tau_{ij} = c_{PI}^{B_j} + d_{pFF} + t_{hold}$$  \hspace{1cm} (15)

for $v_{ij} \in PO_j$

$$\tau_{ij} \leq c_{PO}^{B_j} + C_p - t_{setup}$$  \hspace{1cm} (16)

$$\tau_{ij} \geq c_{PO}^{B_j} + c_{PI}^{B_j} + t_{hold} - d_{pFF}$$  \hspace{1cm} (17)

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

$$\tau_{ij} \geq \tau_{kj} + d_{ij} + \Delta d_{ij} \quad v_{kj} \in FI(v_{ij})$$  \hspace{1cm} (18)

$$\tau_{ij} \leq \tau_{kj} - (d_{kj} + \Delta d_{kj}) \quad v_{kj} \in FO(v_{ij})$$  \hspace{1cm} (19)

where $\tau_{ij}$ is the variable representing both the arrival time and required time of a node $v_{ij}$, $c_{PI}^{B_j}$ is the clock arrival time at the primary input flip-flops of the block $B_j$ and $c_{PO}^{B_j}$ 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 0.13µm 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
The approach is capable of determining potential slacks with significant run-time speedup over the existing methods.

### 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 1: Comparison of MISA, GREEDY and LP

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Number of Gates</th>
<th>Total Slack</th>
<th>Potential Slack</th>
<th>Improve (%)</th>
<th>CPU Time (second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MISA</td>
<td>GREEDY</td>
<td>LP</td>
<td>MISA</td>
<td>GREEDY</td>
<td>LP</td>
</tr>
<tr>
<td>apex7</td>
<td>184</td>
<td>167294</td>
<td>74132</td>
<td>60247</td>
<td>74132</td>
</tr>
<tr>
<td>C432</td>
<td>192</td>
<td>167587</td>
<td>65311</td>
<td>53730</td>
<td>65311</td>
</tr>
<tr>
<td>alu2</td>
<td>283</td>
<td>227729</td>
<td>94902</td>
<td>66307</td>
<td>94902</td>
</tr>
<tr>
<td>C1908</td>
<td>448</td>
<td>515213</td>
<td>113287</td>
<td>76565</td>
<td>113287</td>
</tr>
<tr>
<td>C1355</td>
<td>460</td>
<td>190382</td>
<td>45361</td>
<td>25311</td>
<td>45361</td>
</tr>
<tr>
<td>apex6</td>
<td>510</td>
<td>450577</td>
<td>249662</td>
<td>169604</td>
<td>249662</td>
</tr>
<tr>
<td>dalu</td>
<td>885</td>
<td>2067609</td>
<td>560450</td>
<td>309773</td>
<td>560450</td>
</tr>
<tr>
<td>C5315</td>
<td>1189</td>
<td>1873252</td>
<td>688346</td>
<td>504960</td>
<td>688346</td>
</tr>
<tr>
<td>ex5p</td>
<td>1636</td>
<td>1148174</td>
<td>350928</td>
<td>251733</td>
<td>350928</td>
</tr>
<tr>
<td>C7552</td>
<td>1742</td>
<td>3529573</td>
<td>879996</td>
<td>537157</td>
<td>879996</td>
</tr>
<tr>
<td>i10</td>
<td>1802</td>
<td>4875123</td>
<td>1632618</td>
<td>1356682</td>
<td>1632618</td>
</tr>
<tr>
<td>des</td>
<td>2445</td>
<td>2593867</td>
<td>1197056</td>
<td>752829</td>
<td>1197056</td>
</tr>
<tr>
<td>misex3</td>
<td>2795</td>
<td>1361256</td>
<td>557167</td>
<td>386248</td>
<td>557167</td>
</tr>
<tr>
<td>alu4</td>
<td>2876</td>
<td>1422486</td>
<td>663673</td>
<td>590664</td>
<td>663673</td>
</tr>
<tr>
<td>seq</td>
<td>3393</td>
<td>1837170</td>
<td>792068</td>
<td>669679</td>
<td>792068</td>
</tr>
<tr>
<td>apex2</td>
<td>3891</td>
<td>1866091</td>
<td>751037</td>
<td>658911</td>
<td>751037</td>
</tr>
<tr>
<td>ex1010</td>
<td>4207</td>
<td>2509745</td>
<td>845325</td>
<td>679306</td>
<td>845325</td>
</tr>
<tr>
<td>spla</td>
<td>6906</td>
<td>5096193</td>
<td>1990314</td>
<td>1601929</td>
<td>1990314</td>
</tr>
<tr>
<td>pdc</td>
<td>8857</td>
<td>6158465</td>
<td>2322928</td>
<td>1836508</td>
<td>2322928</td>
</tr>
</tbody>
</table>

![Figure 4: CPU Time vs. Number of gates](image_url)
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


<table>
<thead>
<tr>
<th>Circuit</th>
<th>FF Number</th>
<th>Gate Number</th>
<th>Potential Slack</th>
<th>Area Reduction Percentage after Gate Sizing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Zero Skew</td>
<td>Global Budgeting</td>
</tr>
<tr>
<td>s208</td>
<td>8</td>
<td>104</td>
<td>50231</td>
<td>65278</td>
</tr>
<tr>
<td>s1196</td>
<td>18</td>
<td>529</td>
<td>220039</td>
<td>273512</td>
</tr>
<tr>
<td>s1423</td>
<td>74</td>
<td>657</td>
<td>284617</td>
<td>332479</td>
</tr>
<tr>
<td>s5378</td>
<td>179</td>
<td>2779</td>
<td>1208258</td>
<td>1671921</td>
</tr>
<tr>
<td>s13207</td>
<td>669</td>
<td>7951</td>
<td>1889762</td>
<td>2466810</td>
</tr>
<tr>
<td>s15850</td>
<td>597</td>
<td>9772</td>
<td>2309835</td>
<td>2938763</td>
</tr>
<tr>
<td>s35932</td>
<td>1728</td>
<td>16065</td>
<td>3708499</td>
<td>4480263</td>
</tr>
</tbody>
</table>