Simultaneous Slack Budgeting and Retiming for Synchronous Circuits Optimization

Shenghua Liu1  Yuchun Ma1  Xianlong Hong1  Yu Wang2
Tsinghua National Laboratory for Information Science and Technology
1Department of Computer Science & Technology, 2Department of Electronic Engineering
Tsinghua University, Beijing 100084, China

Abstract - With the challenges of growing functionality and scaling chip size, the possible performance improvements should be considered in the earlier IC design stages, which gives more freedom to the later optimization. Potential slack as an effective metric of possible performance improvements is considered in this work which, as far as we known, is the first work that maximizes the potential slack by retiming for synchronous sequential circuit. A simultaneous slack budgeting and incremental retiming algorithm is proposed for maximizing potential slack. The overall slack budget is optimized by relocating the FFs iteratively with the MIS-based slack estimation. Compared with the potential slack of a well-known min-period retiming, our algorithm improves potential slack averagely 19.6% without degrading the circuit performance in reasonable runtime. Furthermore, at the expense of a small amount of timing performance, 0.52% and 2.08%, the potential slack is increased averagely by 19.89% and 28.16% separately, which give a hint of the tradeoff between the timing performance and the slack budget.

I Introduction

With the coming up of SoC (system on a chip) and SiP (system in a package), more and more devices trend to be put in the small silicon area while at the same time the clock frequency is pushed even higher. Thus timing, area, and power dissipation, which are the three major design objectives, become even more challenging in modern circuit design. Typically, the goal of performance optimization is either to minimize the clock period within a given area and/or power budget [1-3], or to minimize their area and/or power dissipation under timing constraints [4-10]. Particularly, timing budget is often performed in order to slow down as many components as possible without violating the system’s timing constraints. The slowed-down components can be further optimized to improve system’s area, power dissipation, or other design quality metrics.

For timing-constrained gate-level synthesis, time slack is an effective metric of circuit’s potential performance improvement. The slack budgeting problem on a graph has been studied in theory and practice for many different applications, such as timing-driven placement [16-17] and floorplanning [11], gate/wire sizing and power optimization [12-15], and etc. Most of the previous slack budgeting approaches are suboptimal heuristics such as Zero-Slack Algorithm (ZSA) [18]. In [12, 19-20], slack budgeting problem in combinational circuit is formulated as maximum-independent-set (MIS) on sensitive transitive closure graph. Recently, [21] presented an LP-based slack budgeting and maximized the potential slack by clock skew optimization. [22] solved the problem of integral delay budgeting through LP relaxation. [23] proposed combinatorial methods based on net flow approach to handle the slack budget problem. However, the existing slack budgeting algorithms are either used for combinatorial circuit, or limited to fixed FF locations. At the early design stages, there is flexibility to schedule pipeline or timing distribution to obtain more timing slack.

As one of the most powerful sequential optimization techniques, retiming was first proposed by Leiserson and Saxe in 1983 [24], which relocates the flip-flops (FFs) in a circuit while preserving its functionality. The past twenty years saw retiming’s effectiveness on improving circuits’ timing, area, and power characteristics. Recent publications [1] and [3] proposed a very efficient retiming algorithm for minimal period by algorithm derivation. [2] and [5] later presented efficient incremental algorithms for min-period retiming under setup and hold constraints, and min-area retiming under a given clock period. [8, 25-26] proposed the ILP/MILP-based algorithms of retiming for power reduction. Besides, retiming can also increase the potential slack which translates to potential performance. As illustrated in Fig. 1, the period of a circuit is minimized with the delay and slack labeled beside each gate as well. It is seen that there is no potential slack in this circuit. However, if retiming process is taken, i.e. moving the FF from edge ED to DC, the potential slack can be increased from 0 to 4, keeping the period minimized at the same time. Though there are massive retiming algorithms, none of them can take slack budget into consideration directly. [28] proposed an LP formulation for optimizing the slack budgeting on interconnects between gates, in order to guide timing-driven placement. However, such an interconnect-based slack budgeting formulation is not easy to transform to gate-based one, since one gate can belong to different interconnects. Furthermore, solving LP formulation with a large problem size is not efficient. In this paper, we propose an incremental retiming approach which can optimize the slack distribution on gates efficiently without any performance degradation.

Fig. 1. Relocate a FF to increase potential slack without degrading timing performance

As far as we know, this paper is the first work that maximizes the potential slack by retiming for synchronous sequential circuit. A simultaneously slack budgeting and incremental retiming algorithm is proposed. The formulation

---

This work is supported by NSFC 60606007, 60870001, and 60720106003, 863 project (No. 2009AA01Z130), and TNLList Cross-discipline Foundation.
of MIS on sensitive transitive closure graph is extended from combinatorial circuit to synchronous sequential circuit for potential slack estimation. The overall slack budget can be optimized by relocating the FFs iteratively with the MIS based slack estimation.

Experimental results show that comparing with the potential slack of a well-known min-period retiming [3], our retiming approach increases the potential slack averagely by 19.6% without degrading timing performance. Meanwhile, retiming and slack budgeting is performed with relaxation of the timing constraint by 2% and 5%, and the potential slack increases 19.6% and 28.16% respectively, which gives a hint to designers about the tradeoff between performance and slack budget or other metrics.

The rest of this paper is organized as follows. Section II introduces the problem formulation. Section III gives three motivating examples of retiming for maximizing potential slack. Sensitive transitive closure graph and slack budgeting for estimating potential slack of synchronous circuit are described in Section IV. Section V presents the simultaneous slack budgeting and retiming algorithm. Experimental results are given in Section VI, and our paper concludes in Section VII.

II. Preliminaries

As in [24], we model a synchronous sequential circuit as a directed graph \( G = (V, E, d, w) \). Each vertex \( v \in V \) represents a combinational gate and each edge \((u, v) \in E\) represents a signal passing from gate \( u \) to gate \( v \). Non-negative gate delays are given as vertex weights \( d: V \rightarrow R^* \) and non-negative integer \( w: E \rightarrow Z^* \) as the edge weight represents the number of FFs on the signal pass. A special host vertex \( H \), the edges from host to the primary inputs, and the edges from the primary outputs to host, are introduced into the graph to represent interfaces with the external environment. Without loss of generality, we assume that \( G \) is strongly connected. Conventionally, three non-negative labels, \( a_i, \gamma_i, \) and \( s_i \): \( V \rightarrow R^* \), represent the latest arrival time, require time, and slack of gate \( i \). \( a_i \) and \( \gamma_i \) can be calculated recursively.

\[
\begin{align*}
\text{init: } a_i &= d_i & \text{if } w_{m,n} > 0 \text{ or } n \in FO(H) \\
a_i &= \max_j (a_j + d_j) & v_j \in FI(v_i) \\
\text{init: } \gamma_i &= T & \text{if } w_{m,n} > 0 \text{ or } m \in FO(H) \\
\gamma_i &= \min_i (\gamma_i - d_i) & v_i \in FO(v_i)
\end{align*}
\]

where \( w_{m,n} \) means the number of FFs on any directed edge \( (v_m, v_n) \), and \( FI(v_i) \) and \( FO(v_i) \) represent the incoming and outgoing gates to gate \( v_i \) respectively. \textit{init} is the initialization condition. Slack \( s_i \) is then calculated by

\[
s_i = \gamma_i - a_i
\]

To represent the reallocation of FFs in retiming, an integer label \( r: V \rightarrow Z^* \) is introduced to represent how many FFs are moved from the outgoing edges to the incoming edges of each node. When \( r \) is negative value, it means FFs are moved from the incoming edges to outgoing edges. Thus the number of FFs on edge \((v_i, v_j)\) with current label \( r \) is as formula (4).

\[
w_{i,j} + r_j - r_i
\]

Our problem is formulated as following.

\textbf{Problem of simultaneous slack budget and retiming (SSBR): Given a directed graph } \( G = (V, E, d, w) \text{ representing a synchronous sequential circuit, and period constraint } T \text{, we want to find FFs reallocation represented by } r \text{ such that the potential slack } P_s \text{ obtained by slack budgeting is maximized under the period constraint.}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{fig3.png}
\caption{Fig. 3. Three scenarios for improving potential slack by retiming}
\end{figure}

As one of the most powerful sequential optimization techniques, retiming has been used to minimize the clock period, area and power. Besides of those benefits, retiming can be used as an effective optimization method to improve the potential slack while preserving circuits’ functionality and timing performance. By studying the topological structure of circuits, we find three different scenarios that the potential slack can be improved significantly by relocating the FFs.
the FFs.

**Scenario I**: make use of “slack” on an output. As shown in Fig. 3 (a), there is no slack for each gate. But after further analysis, the signal arrival time of the output edge \((v_1, v_2)\) of gate \(v_1\) is 2. And this output signal is actually required to arrive at the input of the FF on edge \((v_1, v_2)\) before 10. Thus the timing “slack” on this output is 8. Since the signal arrival time of the other output \((v_1, v_3)\) is required before \(2 - d_2 = 2\), according to formulas (2) and (3), the required time on this gate is \(\gamma_1 = 2\), and slack is \(s_1 = 0\). However, with moving FF from \((v_1, v_3)\) to \((v_1, v_4)\), the timing “slack” on the output can be used by the gates \(v_1\) and \(v_4\), as Fig. 3 (b) shows. So, the total potential slack increased from 0 to 8.

**Scenario II**: redistribute the slack to the sub-graph with more branches. The sub-graph here refers to one of the sub-graphs divided by FFs in graph \(G\). The branches are actually the independent set, in which vertices are not slack sensitive to each other (Explained in Section IV). As Fig. 3 (c) shows, set \(\{v_1, v_2\}\) is an independent set in sub-graph of \(\{v_1, v_2, v_3, v_5, v_6\}\), while the size of independent set in other part is 1. Thus when moving the FF from edge \((v_4, v_5)\) to \((v_3, v_5)\) as in Fig. 3 (d), slack is increased from 2 to \(s_2 + s_6 = 3\).

**Scenario III**: move FF to the gate side with more edges. It is obvious that such a movement increases the slack at least \(2d_i\) shown in Fig. 3 (e) and (f). However, such potential slack is increased at the expense of more chip area and power consumed by more FFs. Thus, a combined cost to estimate the real potential performance is given as follows.

\[
\mathcal{P}^* = \mathcal{P} - \alpha \Delta n
\]

where \(\mathcal{P}\) and \(\mathcal{P}^*\) are potential slack and combined potential slack, considering the cost of increasing FFs. \(\Delta n\) is the increased number of FFs, and constant \(\alpha\) reflects the impacts of add one more FFs to the circuits. For the purpose of maximizing the potential slack, \(\alpha\) can be assigned to the delay of a FF. And if the chip area is very critical, \(\alpha\) can be a very large number, so that **Scenario III** will not be considered.

It is worth to mention that max-potential-slack retiming may become complicated when there appears any negative-weight edge, after relocating FFs. For such case, we propose a more general method in Section V. Thus, to solve the SSBR problem, it not only needs slack budgeting for various circuit topologies affected by different FF locations, but also needs iteratively retiming to find a optimized FF locations that maximizes potential slack.

IV. Time Slack Budgeting for Estimating Potential Slack

To estimate the potential slack, a sensitive transitive closure graph is built up for a synchronous circuit. And a MIS-based method is presented for budgeting slack.

**A. Sensitive transitive closure graph**

The essential difference between combinatorial circuits and synchronous sequential ones is that the later ones have FFs on signal passes. The edge with FFs cannot be slack sensitive because the signals on the two sides of FFs are in different stage of sequential circuits. Thus, we could easily borrow the concepts of slack sensitive from combinatorial circuit with some modifications. With the help of the labels \(\alpha/\gamma/s\) and \(r\) on each vertex of graph \(G (V, E, d, w)\), we formally define edge \((v_i, v_j) \in E\) to be slack sensitive if and only if the condition (6) is satisfied.

\[
w_i + r - r_j = 0 \quad \text{and} \quad (a_j - a_i = d_i \quad \text{or} \quad \gamma_i - \gamma_j = d_j)
\]

where \(w_i + r - r_j = 0\) means there is no FF on edge \((v_i, v_j)\). From formula (1), the edge with \(a_j - a_i = d_i\) is a sensitive edge, on which the arrival time \(a_i + d_i\) of gate \(v_i\) actually depends. In the same way, from formula (2), the edge with \(\gamma_i - \gamma_j = d_j\) of gate \(v_i\) depends on it. According to the definition, for the example in Fig. 3 (d), edge \((v_1, v_2)\) is not slack sensitive \((r_1 = r_2 = 0, \alpha_2 - \alpha_1 = d_i\) and \(\gamma_2 - \gamma_1 = d_i\), while edge \((v_3, v_2)\) is slack sensitive, \((r_2 = r_3 = 0, \alpha_2 - \alpha_3 = d_3)\). The sensitive transitive closure graph (STCG) is \(G(V, E)\) of \(G\) is a directed graph such that there is an edge \((v_i, v_j)\) in \(G\), if and only if there is a directed path from \(v_i\) to \(v_j\) in \(G\) and the path consists of only sensitive edges. Two vertices \(v_i\) and \(v_j\) are called slack insensitive if there is no edge from \(v_i\) to \(v_j\) or from \(v_j\) to \(v_i\) in \(G\). A set \(SI = \{v_1, v_2, ..., v_j\}\) of vertices is called slack insensitive if all vertices in the set are pair-wise slack insensitive. Let each edge in \(G\), replaced by an undirected edge, and become an undirected graph \(\tilde{G}_s\). So, the slack insensitive set \(SI\) is an independent set of graph \(\tilde{G}_s\). Fig. 4 (a) highlights the slack sensitive edges in Fig. 3 (d) with dotted lines. And then the undirected STCG is given in Fig. 4 (b) which consists of two connected sub-graphs.

**B. Estimating Potential Slack (EPS)**

After labeling \(\alpha/\gamma/s\), we collect all the vertices with positive slack into set \(Q\). An induced graph of \(\tilde{G}_s\) on \(Q\) is named as \(\tilde{G}_Q\). And let a maximum independent set of \(\tilde{G}_Q\) be \(SI\). Then, each vertex in set \(Q\) can increase its delay by \(S = \min\{s_i \mid v_i \in SI\}\) separately without breaking the clock period constraint. And then we iteratively find the set \(SI\), and increase delay until \(Q\) is empty. At last, the potential slack is the summation of the increased delay in every iteration. The algorithm of potential slack budgeting is given in Fig. 5. The optimality of the algorithm was proved in [12]. And a well-know polynomial-time MIS solver by Dharwadker [27] is employed in step 4 of Fig. 5, which solves the MIS problem by the duality with minimum vertex cover.

We still take the example in Fig. 4 to illustrate the process. The induced graph of \(\tilde{G}_s\) on \(\{v_1, v_2, v_3, v_5, v_6\}\) is the left connected sub-graph. Thus the maximum independent set \(SI\) is \(\{v_2, v_6\}\), \(S = 1\), and \(p = 2\). After increasing the delay of each vertex, \(\tilde{G}_s\) contains only vertex \(v_2\) with slack \(s_2 = 1\). Thus \(SI\)
= \{v_1\}, S=1 and p_1=2+ S=3. Afterwards, the subroutine terminates and successfully output the budgeted slack p_s = 3.

1. p_s = 0;
2. let vertices with positive slack be Q;
3. While (Q is nonempty) {
   4. build induced graph of \(\sigma_s\) on Q;
   5. find maximum independent set SI;  
   6. \(S = \min\{s_j | v_j \in SI\}\);
   7. \(p_s += S \times \text{sizeof}(SI)\);
   8. Foreach \(v_i \in Q\) \(\{d_i += S; \}\)
   9. update \(a/\gamma/s\) and Q;
   10. }
11. recover delay \(d\) of graph \(G\);
12. output potential slack \(p_s\);

Fig. 5. Subroutine for estimating potential slack (EPS)

V. Simultaneous Retiming and Slack Budgeting

A. General form for SSBR

In section III, we enumerate three scenarios that improve the potential slack by relocating FFs. However, for the case shown in Fig. 3 (a), if retiming vertex \(v_i\) with labeling \(r_i = 1\) without considering clock period (Fig.6(a)), the new weight of edge \((v_1, v_2)\) is \(w_{v_2} + r_2 - r_1 = -1\), according to formula (4). It means that the edge \((v_1, v_2)\) needs to borrow a FF from outgoing edges of vertex \(v_2\), in order to legalize the retiming. Thus with \(r_2 = 1\), the retiming result is as Fig. 6(b) shows. This procedure is called retiming legalization. Such retiming legalization usually causes the relocation of the FFs in other part of the circuit which may in turn reflect the slack distribution. So, in this section, a more general method is proposed to increase potential slack with retiming under the specific period constraint.

Thus simultaneous slack budgeting and retiming can be expressed in the following form.

\[
\begin{align*}
\text{Max} & \quad p_s \\
\text{s.t.} & \quad w_{ij} + r_i - r_j \geq 0, \forall (v_i, v_j) \in E \quad (8) \\
& \quad a_i \leq T, \forall v_i \in V \quad (9)
\end{align*}
\]

Formula (8) shows the constraints that the number of FFs on each edge should not be negative. Formula (9) constrains that the delay of critical path must satisfy the timing constraint \(T\).

B. Incremental retiming with possible moves

Therefore, we can relocate the positions of FFs iteratively without violating the timing constraint and estimate the potential slack for each relocation to find an optimal one. Each step is triggered by a FF relocation which satisfies the timing constraints. Let such an FF relocation be a movement \(m_i\) represented by label \(r\). Firstly, we define movement \(m_i\) over vertex \(v_i\) is positive if only if it moves FFs from outgoing edges \(FO(v_i)\) of \(v_i\) to its incoming edges \(FI(v_i)\), and the reverse movement is negative. If the movement \(m_i\) is positive, the delay \(d_i\) of \(v_i\) will be added to the timing path starting with gate \(v_i\) by formula (10). Thus, in order to keep condition (10) satisfied, it is necessary that the slack \(s_j\) of gate \(v_j\) must be no smaller than the delay \(d_i\). Hence, condition (10) must be satisfied.

\[
\gamma_j - d_j - d_i \geq 0, \quad v_j \in FO(v_i) \quad (10)
\]

In the same way, if the movement \(m_i\) is negative, the delay \(d_i\) of \(v_i\) will be added to the timing path ending with gate \(v_j \in FI(v_i)\). To keep condition (9) satisfied, it is necessary that the arrival time \(a_i\) of gate \(v_i\) plus the delay \(d_i\) must not exceed the period \(T\). That is condition (11) must be satisfied.

\[
a_i + d_i \leq T, \quad v_j \in FI(v_i) \quad (11)
\]

So, we can collect the possible FF movements \(M\) with conditions (10) or (11) satisfied. If condition (10) is satisfied, the next movement \(m_i \in M\) is possible and represented by label \(r\) as \(m_i = r_i + 1\), otherwise if condition (11) is satisfied, it is \(m_i = r_i - 1\). Although conditions (10) and (11) can reduce much searching space, they are insufficient for condition (8), so some of them have to be picked out after retiming legalization and arrival time calculation, which are discussed later.

When relocating FFs by positive movement \(m_i \in M\), some outgoing edges may become negative weighted. As described in subsection A, the retiming legalization may borrow the FFs from the outgoing edges of the succeeding gates in the directed path. It is practiced recursively by invoking formula (12), until all edges weight are positive.

\[
r_j = r_i - w_{ij}, \quad v_j \in FO(v_i), \quad w_{ij} > r_j - r_i < 0 \quad (12)
\]

In the same way, when relocating FFs by negative movement \(m_i\), if there occur negative weighted edges, we legalize the retiming by borrowing the FFs from the incoming edges of the preceding gates. And the retiming legalization is based on formula (13) recursively.

\[
r_j = r_i + w_{ij}, \quad v_j \in FI(v_i), \quad w_{ij} > r_i - r_j < 0 \quad (13)
\]

After the retiming legalization, the arrival time of each gate is calculated by formula (1). If condition (9) is not satisfied, such movement \(m_i\) will not be considered. If satisfied, an incremental retiming is performed based on the movement \(m_i\).

1. \(M = \varnothing, P_s = 0;\)
2. Do {
   3. collect possible movements \(M\) by (10) and (11);
   4. isImproved = false;
   5. Foreach \(m_i \in M\) {
      6. relocate FFs by movement \(m_i\);
      7. retiming legalization by (12) and (13);
      8. calculate arrival time \(a\) by formula (1);
      9. if (condition (9) is unsatisfied) continue;
      10. calculate labels \(\gamma\) and \(s\) by (2) and (3);
      11. estimate potential slack \(p_s\) by EPS;
      12. if \((P_s < p_s)\) {
         13. update retiming label \(r\);
         14. isImproved = true;
         15. \(P_s = p_s;\)
         16. }
   17. } While (isImproved)
   18. output the maximized potential slack \(P_s\);
   19. output the retiming result represented by label \(r\);

Fig. 7. The algorithm of simultaneous slack budgeting and retiming

C. Iterative incremental retiming flow

The subroutine of EPS is called to estimate potential slack after each incremental retiming with those considered
movements in $M$. Only the movement that improves the potential slack is accepted, and then retiming label $r$ is updated. The algorithm iteratively collects possible FF movements $M$, and takes the movement that can increase potential slack. The algorithm terminates until there are no potential slack increase any more, and its flow is illustrated in Fig. 7.

![Image](66x474 to 282x664)

Fig. 8. An example of our incremental retiming and slack budgeting algorithm. The movements are labeled besides related vertex shown in (b) and (c).

An example in Fig. 8 is used to clarify our idea of incremental retiming for maximizing potential slack. Detailed execution results are listed in the following table step by step, where $M$ is the movements set represented by $r$, $S_I$ is the maximum independent set of $G_S$ with positive slack vertices, and $P_r$ is the potential slack. At last, the potential slack increases from 3 to 8.

<table>
<thead>
<tr>
<th>$M$, $S_I$, $P_r$</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 $\varphi, {v_4, v_6, v_8}$, 3</td>
<td>$r_c=0$, $v_i \in V$</td>
</tr>
<tr>
<td>2 ${r_c+1, r_c+1}, {v_4, v_6, v_8}$, 3</td>
<td>after step 3 in Fig. 7</td>
</tr>
<tr>
<td>3 ${r_c+1, r_6+1}$, ${v_6, v_8, v_{10}}$, 4</td>
<td>taking movement $r_c+1$, and $r_6+1$</td>
</tr>
<tr>
<td>4 ${r_c+1, r_6+1}, {v_6, v_8, v_9}$, 8</td>
<td>taking movement $r_c+1$, and $r_6+1$, $r_9+1$</td>
</tr>
<tr>
<td>5 ${r_c+1, r_6+1}, {v_6, v_8, v_{10}}$, 8</td>
<td>after step 3 in Fig. 7</td>
</tr>
<tr>
<td>6 $\varphi, {v_6, v_8, v_{10}}$, 8 (optimal)</td>
<td>condition (10) is unsatisfied, and $istImproved$ is false.</td>
</tr>
</tbody>
</table>

VI. Experimental Results

The algorithm is implemented with C++, and tested under linux server with eight 3.0-GHz cores and 6G memories. 19 cases from the ISCAS89 benchmarks are tested, and the name, number of gates, number of signal passes, the maximum number of gate outputs/inputs, and the minimum period for each case are separately given in Table 1.

In the experiments, a min-period retiming algorithm [3] is employed to get the minimum period of the circuits, and potential slacks are estimated under current fixed FFs, which are listed in the 2nd and 3rd column of TABLE II. The simultaneous slack budgeting and retiming are firstly performed on those test cases within min-period timing constraint. The maximized potential slack, the increase ratio of potential slack, and runtime are given separately. The average increase ratio of potential slack is 19.6% within timing constraint of the min-period $T_{min}$ in reasonable runtime. And then we relax the timing constraint and perform the algorithm within timing constraint of (1+2%) $T_{min}$ and (1+5%) $T_{min}$. Since the clock period may not reach the bound of the timing constraint, the actual clock period after retiming, the potential slack, the slack increase ratio compare with $P_0$, and runtime are given separately for both experiments with timing constraint relaxation in TALBE II. The average increase ratios are 19.89% and 28.16% with a little sacrifice of timing performance (increasing the average period by 0.52% and 2.08%), which can give designers some hints to the tradeoff between fast and low-power circuits. Fig. 9 illustrates a direct comparison between $P_0$ and the maximized potential slack within three different timing constraints which is normalized by being divided by $P_0$. Since the room of slack optimization with retiming depends on the topology of circuit greatly, the improvement varies from 0 to 214.29%. As for s27.test, the circuit size is small, so the increase ratio appears relatively high. We can see that in addition to s27.test, some cases also achieve much improvement, such as 58.76% and 20.88%, while there is no improvement at all for some others. Furthermore, the relaxed timing constraints may even not influence slack budget. Averagely, we can obtain about 9% extra slack budget (19.6% vs. 28.16%) with only 2% performance degradation (80.95 vs. 82.63).

![Image](329x130 to 551x270)

**TABLE I**

The Characteristics of Test cases

<table>
<thead>
<tr>
<th>Test case name</th>
<th>Gates</th>
<th>Edges</th>
<th>Max. outputs</th>
<th>Max. inputs</th>
<th>$T_{min}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>s27.test</td>
<td>11</td>
<td>19</td>
<td>4</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>s208.1.test</td>
<td>105</td>
<td>182</td>
<td>19</td>
<td>4</td>
<td>28</td>
</tr>
<tr>
<td>s298.test</td>
<td>120</td>
<td>250</td>
<td>13</td>
<td>6</td>
<td>24</td>
</tr>
<tr>
<td>s382.test</td>
<td>159</td>
<td>312</td>
<td>21</td>
<td>6</td>
<td>44</td>
</tr>
<tr>
<td>s380.test</td>
<td>160</td>
<td>354</td>
<td>36</td>
<td>7</td>
<td>64</td>
</tr>
<tr>
<td>s344.test</td>
<td>161</td>
<td>280</td>
<td>12</td>
<td>11</td>
<td>46</td>
</tr>
<tr>
<td>s349.test</td>
<td>162</td>
<td>284</td>
<td>12</td>
<td>11</td>
<td>46</td>
</tr>
<tr>
<td>s444.test</td>
<td>182</td>
<td>358</td>
<td>22</td>
<td>6</td>
<td>46</td>
</tr>
<tr>
<td>s526.test</td>
<td>194</td>
<td>451</td>
<td>13</td>
<td>6</td>
<td>42</td>
</tr>
<tr>
<td>s526m.test</td>
<td>195</td>
<td>451</td>
<td>13</td>
<td>6</td>
<td>42</td>
</tr>
<tr>
<td>s510.test</td>
<td>212</td>
<td>431</td>
<td>28</td>
<td>7</td>
<td>50</td>
</tr>
<tr>
<td>s420.1.test</td>
<td>219</td>
<td>384</td>
<td>31</td>
<td>4</td>
<td>40</td>
</tr>
<tr>
<td>s832.test</td>
<td>288</td>
<td>788</td>
<td>107</td>
<td>19</td>
<td>98</td>
</tr>
<tr>
<td>s820.test</td>
<td>290</td>
<td>776</td>
<td>106</td>
<td>19</td>
<td>92</td>
</tr>
<tr>
<td>s641.test</td>
<td>380</td>
<td>563</td>
<td>35</td>
<td>24</td>
<td>238</td>
</tr>
<tr>
<td>s713.test</td>
<td>394</td>
<td>614</td>
<td>35</td>
<td>23</td>
<td>262</td>
</tr>
<tr>
<td>s838.1.test</td>
<td>447</td>
<td>788</td>
<td>55</td>
<td>4</td>
<td>80</td>
</tr>
<tr>
<td>s1238.test</td>
<td>509</td>
<td>1055</td>
<td>192</td>
<td>14</td>
<td>110</td>
</tr>
<tr>
<td>s1488.test</td>
<td>654</td>
<td>1406</td>
<td>56</td>
<td>19</td>
<td>166</td>
</tr>
</tbody>
</table>

Fig. 9. Normalized $P_0$ and the optimized potential slack under three different timing constraints.

VII. Conclusions

In this work, we introduce the slack sensitive closure graph in combinatorial circuits to synchronous sequential circuits, on which a MIS-based slack budgeting is performed.
And a simultaneously incremental retiming and slack budgeting algorithm, i.e. max-potential-slabk retiming is presented, which is heuristic yet effective. Comparing with the potential slack obtained by a min-period retiming [3], the experimental results show that our algorithm can improve the potential slack averagely by 19.6% without degrading timing performance in reasonable runtime. Meanwhile, we also perform max-potential-slabk retiming with timing constraint relaxation of 1% and 2%, and the potential slack increases 19.6% and 28.16%, which gives a view of tradeoff between a fast and low-power design. Since potential slack is a good prediction for potential performance in the gate-level synthesis [19], the retiming for power reduction maximization is our future work to practice its effectiveness.

References


<table>
<thead>
<tr>
<th>Test cases</th>
<th>Zhou’s [3]</th>
<th>Under T&lt;sub&gt;min&lt;/sub&gt; constraint</th>
<th>Relaxing 2%</th>
<th>Relaxing 5%</th>
</tr>
</thead>
<tbody>
<tr>
<td>T&lt;sub&gt;min&lt;/sub&gt;</td>
<td>P&lt;sub&gt;j&lt;/sub&gt;</td>
<td>P&lt;sub&gt;j&lt;/sub&gt; increase</td>
<td>runtime(s)</td>
<td>period</td>
</tr>
<tr>
<td>s27.test</td>
<td>20 14</td>
<td>44 214.29%</td>
<td>0.00</td>
<td>20 44 214.29%</td>
</tr>
<tr>
<td>s208.1.test</td>
<td>28 422</td>
<td>480 13.74%</td>
<td>1.30</td>
<td>28 480 13.74%</td>
</tr>
<tr>
<td>s298.test</td>
<td>24 522</td>
<td>628 20.31%</td>
<td>6.64</td>
<td>24 628 20.31%</td>
</tr>
<tr>
<td>s382.test</td>
<td>44 1204</td>
<td>1204 0.00%</td>
<td>11.58</td>
<td>44 1204 0.00%</td>
</tr>
<tr>
<td>s386.test</td>
<td>64 724</td>
<td>724 0.00%</td>
<td>1.11</td>
<td>64 724 0.00%</td>
</tr>
<tr>
<td>s344.test</td>
<td>46 724</td>
<td>1178 58.76%</td>
<td>8.19</td>
<td>46 1178 58.76%</td>
</tr>
<tr>
<td>s349.test</td>
<td>46 728</td>
<td>800 20.88%</td>
<td>7.72</td>
<td>46 800 20.88%</td>
</tr>
<tr>
<td>s444.test</td>
<td>44 1208</td>
<td>1264 6.46%</td>
<td>33.55</td>
<td>44 1264 6.46%</td>
</tr>
<tr>
<td>s526.test</td>
<td>42 1324</td>
<td>1472 11.18%</td>
<td>27.08</td>
<td>42 1472 11.18%</td>
</tr>
<tr>
<td>s528.test</td>
<td>42 1326</td>
<td>1466 10.72%</td>
<td>57.80</td>
<td>42 1466 10.72%</td>
</tr>
<tr>
<td>s510.test</td>
<td>50 1204</td>
<td>1224 0.00%</td>
<td>12.21</td>
<td>50 1224 0.00%</td>
</tr>
<tr>
<td>s420.1.test</td>
<td>40 1318</td>
<td>1444 9.56%</td>
<td>42.68</td>
<td>40 1444 9.56%</td>
</tr>
<tr>
<td>s832.test</td>
<td>98 6130</td>
<td>6130 0.00%</td>
<td>60.01</td>
<td>98 6130 0.00%</td>
</tr>
<tr>
<td>s830.test</td>
<td>92 5746</td>
<td>5746 0.00%</td>
<td>42.24</td>
<td>92 5746 0.00%</td>
</tr>
<tr>
<td>s641.test</td>
<td>238 9624</td>
<td>9624 0.00%</td>
<td>329.63</td>
<td>238 9624 0.00%</td>
</tr>
<tr>
<td>s713.test</td>
<td>262 11224</td>
<td>11760 4.68%</td>
<td>431.90</td>
<td>262 11760 4.68%</td>
</tr>
<tr>
<td>s838.1.test</td>
<td>80 7038</td>
<td>7292 3.61%</td>
<td>1840.07</td>
<td>80 7292 3.61%</td>
</tr>
<tr>
<td>s1238.test</td>
<td>110 9260</td>
<td>9260 0.00%</td>
<td>3318.06</td>
<td>110 9462 2.18%</td>
</tr>
<tr>
<td>s8488.test</td>
<td>166 14346</td>
<td>11436 0.00%</td>
<td>884.97</td>
<td>166 11462 1.98%</td>
</tr>
</tbody>
</table>

Average | 81.95 | 19.6% | 165.93 | 81.71(±0.25%) | 19.89% | 427.54 | 82.43(±2.08%) | 28.16% | 1027.49 |