Performance-Driven Partitioning using a Replication Graph Approach

Lung-Tien Liu, Ming-Ter Kuo, Chung-Kuan Cheng, and Te C. Hu

Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093-0114

Abstract—An efficient algorithm is proposed to tackle the performance-driven partitioning problem using retiming and replication. We devise a replication graph to model the composite effect of replication and retiming. With the replication graph, we formulate the problem as an integer linear programming problem. A heuristic algorithm is derived to solve the problem by exploring the dual program of its linear programming relaxation.

1 Introduction

Due to the physical geometric distance and interface technology limitations, intermodule delay is contributing the dominant portion of signal propagation delay. Consequently, we should take into account the intermodule delay for performance-driven partitioning.

For partitioning problem with timing and size constraints, Shih et al. [12, 13] propose a algorithm to guarantee that the delay between registers satisfies the timing constraint. In [9, 10], Liu et al. propose the partitioning algorithms using a retiming technique [6] to explore the ultimate clock period of the circuit.

As mentioned in [9], replication [5, 4] on parts of the circuit can improve the clock period of the partition. However, the combination of replication and retiming complicates the partitioning problem. No systematical solution has been proposed yet.

In this paper, an efficient algorithm is proposed to tackle the performance-driven partitioning problem using retiming and replication.

2 Statement of Problem

In this section, we first take an example to show that replication can improve the crossing edge count and the clock period simultaneously. Then we introduce the definitions and state the performance-driven partitioning problem.

2.1 Motivation of This Paper

Fig. 1 shows a circuit of six combinational elements (in circles) and five registers (in rectangles). Each combinational element has its delay and size, while all registers have zero delays and sizes. Each combinational element is labeled with a delay \(d\). As in [9, 10], we assume that the combinational blocks are fine-grained and therefore can be split or merged. We also assume that each register in the circuit has a single input and a single output since a physical circuit can be transformed into this way as shown in section 8 of [6].

32nd ACM/IEEE Design Automation Conference ©
Permission to copy without fee all or part of this material is granted, provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission. © 1995 ACM 0-89791-756-1/95/0006 $3.50
2.2 Definitions

Data Flow Graph: We use a directed data flow graph \( G = (V, E) \) to represent a circuit, where \( V \) and \( E \) model the combinational elements and the interconnections, respectively. Each node \( i \) is associated with a size \( s_i \). By distributing the combinational delay of node \( i \) into the output edges of \( i \), each edges \( (i, j) \) have a delay time \( d_{ij} \). For a node with no output edges, the delay is placed on its input edges. Each edge \( (i, j) \) has a register count \( r_{ij} \) which denotes the number of registers on the interconnections from combinational element \( i \) to \( j \). Furthermore, each edge \( (i, j) \) is associated with a capacity \( c_{ij} \) representing the number of interconnections from node \( i \) to \( j \).

In this paper, we focus on two-way partitioning. We use function \( \text{size}(X) \) to denote the total size of a node set \( X \). The upper size limits of two partitions are denoted by \( Z_1 \) and \( Z_2 \), respectively.

Partitioning with replication is strongly related to the directions of edges. For two disjoint node sets \( X \) and \( Y \), we use \([X \rightarrow Y]\) to denote the directed cut set from \( X \) to \( Y \). Therefore, \([X \rightarrow Y]\) contains all the cut edges \( (i, j) \) such that \( i \in X \) and \( j \in Y \). We use function \( \text{cut}(X \rightarrow Y) \) to denote the total capacities of the edges in \([X \rightarrow Y]\).

Retiming: Given a data flow graph \( G = (V, E) \), a retiming [6] specifies a transformation of the original graph in which registers are added and removed. Lerserson et al. [6] propose an \( O(nm \log n) \) algorithm to determine the minimum clock period achieved by adopting retiming, where \( n \) and \( m \) denote the numbers of nodes and edges, respectively.

Iteration Bound: While retiming can reduce the clock period of a circuit, there is a lower bound imposed by the feedback loops in the data flow [11]. Given a feedback loop \( l \), let \( d_l \) and \( r_l \) be the sum of edge delays and the sum of registers on loop \( l \), respectively. The delay-to-register ratio of \( l \) is equal to \( \frac{d_l}{r_l} \). The iteration bound \( J \) is defined as the maximum delay-to-register ratio, i.e.,

\[
J = \max \left( \frac{d_l}{r_l} \right) \quad \forall \, \text{loop } l .
\]

Note that if loop \( l \) contains cut edges in a partition, the total edge delay has to include the cut delays.

Cycle Mean Problem: For the special case that each edge has exactly one register on it, the iteration bound problem becomes a cycle mean problem [1]. In our test cases, each combinational block node connects between two register nodes. We can identify the iteration bound and the edges that contribute to the bound in \( O(nm) \) time.

Timing Constraint: We use the iteration bound constraint as the timing constraint in this paper. We have two reasons to use the iteration bound. (i) It is faster to calculate the iteration bound. (ii) The iteration bound stands for the lower bound of the clock period can be achieved by retiming.

2.3 Problem Formulation

Given a data flow graph \( G = (V, E) \) and three node sets \( S, R \) and \( T \) such that \( S \cap T = \emptyset \) and \( R = V - S - T \) as shown in Fig. 2(a), Fig. 2(b) presents a partitioning with \( R \) being the set of replicated nodes. Each copy of \( R \) needs to collect a complete set of input signals.

Given two disjoint sets \( S \) and \( T \), let a replication cut \([S, T]\) denote the cut set of a partitioning with \( R = V - S - T \) being duplicated. From Fig. 2(b), we can see that replication cut \([S, T]\) is the union of four directed cuts,

\[
[S, T] = [S - T] \cup [T - S] \cup [S - R] \cup [T - R].
\]

Figure 3: Two directed cuts

Interpretation of the Cut Set: Suppose we rewrite the replication cut in the format:

\[
[S, T] = [S - \tilde{S}] \cup [T - \tilde{T}]
\]

where \( \tilde{S} \) and \( \tilde{T} \) denote the complementary sets of \( S \) and \( T \), respectively. We can interpret the cut set of the replication cut \([S, T]\) as two directed cuts on the original graph \( G \) as shown in Fig. 3.

Time-Constrained Replication Cut Problem: Given a directed graph \( G \) and a ratio \( K \), find a replication cut \([S, T]\) with an objective

\[
\min \, c([S, T]) = c([S - \tilde{S}] \cup [T - \tilde{T}])
\]

subject to the size constraints that

\[
\text{size}(S \cup R) \leq Z_1, \text{ and size}(T \cup R) \leq Z_2,
\]

and the feasible constraints

\[
S \cap T = \emptyset, R = V - S - T
\]

and the timing constraint

\[
\text{iteration bound of } [S, T] \leq K.
\]

3 Investigation of the Problem

In this section, we first show that the time-constrained replication cut problem without size constraint is \( \mathcal{NP} \)-Hard. Then, the edge dependency problem caused by replication in retiming is illustrated.

By reducing from 3SAT, we have the following theorem [7]:

**Theorem 1:** The time-constrained replication cut problem without size constraint is \( \mathcal{NP} \)-Hard.

Edge Dependency of Replication in Retiming: Due to the replication, the loops which entirely locate in the set
of nodes $S \cup R$ or $T \cup R$ will not be cut. Only those loops which intersect both node sets $S$ and $T$ will be cut. We use Fig. 4 to explain this concept. Given a graph $G$ and a replication cut $[S, T]$ as shown in Fig. 4(a), graph $G^d$ in Fig. 4(b) represents the graph with node set $R$ duplicated. Due to the replication, directed edge $(e, a)$ in $G^d$ emanates from $R$ to $S$ on the same side of the partition and thus does not cross the cut. Similarly, directed edge $(b, c)$ in $G^d$ emanates from $R$ to $T$ on the same side of the partition and thus does not cross the cut.

The loop $(a, d, e, a)$ in $G$ entirely locate in the set of nodes $S \cup R$. So the corresponding loop $(a, d, e, a)$ in $G^d$ will not be cut. On the other hand, the corresponding loop $\ell = (a, b, c, e, a)$ in $G^d$ of the loop $(a, b, c, e, a)$ in $G$ will be cut as shown in Fig. 4(b). Therefore, edges $(a, b)$ and $(c, e)$ contribute cut delay $\delta$ to loop $\ell$ in $G^d$ with respect to replication cut $[S, T]$. These two cut delays result in loop $\ell$ with greater delay-to-register ratio for retiming. Thus, given an edge, the edge delay contributed by the replication cut depends on the configuration of the loop which contains the edge. This is called the edge dependency problem, which is caused by the composite effect of replication and retiming.

4 Construction of the Replication Graph

We devise a replication graph to tackle the edge dependency problem. We assume two nodes $s$ and $t$ are to be separated by the replication cut.

Figure 5: The replication graph $G^\ast$

Construction of Replication Graph: Given the graph $G = (V, E)$ and a pair of nodes $s$ and $t$, we create another graph $G' = (V', E')$ where each node $i'$ corresponds to a node $i$ in $G$ and the directed edge $(j', i')$ in $G'$ is just the reverse of its corresponding edge $(i, j)$ in $G$ with equal capacity (i.e. $c_{j'i'} = c_{ij}$), equal delay time (i.e. $d_{j'i'} = d_{ij}$) and equal register count (i.e. $r_{j'i'} = r_{ij}$). From every node in $G$, we add a directed edge with infinite capacity, zero delay and zero register count to the corresponding node in $G'$. Then, we create a super source $s'$ and a super sink $t'$ which connect nodes $s$, $s'$, $t$ and $t'$ with an infinite capacity, a delay time 0 and a register count 0. We refer to the combined graph as the replication graph $G^\ast = (V^\ast, E^\ast)$ shown in Fig. 5.

Given a directed cut $C = [(i' \in X) \cup (i' \in \bar{X}) \rightarrow (i' \in X') \cup (i' \in \bar{X'})]$ of $G'$ with $V' = X \cup \bar{X}$ and $V' = X' \cup \bar{X'}$, a replication cut $[S, T]$ of the original graph with $S = X, T = \{i | i' \in \bar{X'}\}$ and $R = V - S - T$ is derived. Note that $T$ is derived from the cut in node set $V'$.

5 Heuristic Algorithm

In this section, we incorporate the size constraint and release the requirement that two specified nodes $s$ and $t$ need to be separated.

Since the replication graph can solve the edge dependency problem, we can formulate the time-constrained replication cut as an integer linear programming formulation by utilizing the proposed replication graph. The relaxation of integer constraint derives a lower bound solution. The duality of the relaxed programming problem motivates the concept of penalty function of edge capacities which is the base of the proposed heuristic algorithm of this section. The detail derivation appears in [7].
Fig. 7 shows the outline of the heuristic algorithm Time-Constrained Partitioning using Replication Graph (TPRG). In each iteration, the algorithm identifies the loop which violates the timing constraint and adds penalty to the edges of the loop. The penalty function of edge capacities enforces the following directed cut to avoid cutting through the violating loops. The iteration stops if a feasible solution is found or the number of iterations is beyond the limit.

5.1 Revision of the Replication Graph

We release the constraint of two special nodes $s$ and $t$ in $G$. Therefore, there is no need to create $s^*$ and $t^*$ and their connecting edges in $G^*$.

5.2 Directed Cut using FM Method

We use a directed FM [4] to get a directed cut of $G^*$. However, instead of applying a directed FM method to the original graph as in [4], our approach applies the directed FM to the proposed replication graph to minimize the replication cut cost.

5.3 Penalty Function of Edge Capacities

Let $C^k = [X \cup X' \rightarrow \hat{X} \cup \hat{X}']$ denote the directed cut we found. Based on cut $C^k$, the cut edges and backward edges are assigned cut delay $\delta$ and $-\delta$, respectively. Then, we use the mean cycle algorithm [1] to calculate the iteration bound $h^k$ of $G^*$. In the mean cycle algorithm, we can also compute a value $h_{uv}^k$ for each edge $(u, v)$ in $G^*$, where $h_{uv}^k$ represents the maximum delay-to-register ratio of all loops passing edge $(u, v)$ in $G^*$ with respect to cut $C^k$. If $h_{uv}^k$ is not greater than $K$, the algorithm stops. Otherwise, the capacity of each edges $(u, v)$ in $G^*$ with $h_{uv}^k > K$ is updated as follows:

$$c_{uv}^{k+1} = \max \left\{ 0, c_{uv}^k + \frac{h_{uv}^k - K}{\sum_{(i,j) \in E^* \text{ and } h_{ij}^k > K} (h_{ij}^k - K)} \right\},$$

where $\Delta$ is a given constant. By updating the capacities, edges located on the most critical loops (i.e., their delay-to-register ratio violate the timing constraint most) will get largest penalties. In our program, we set $\Delta$ to be 100.

6 Experimental Results

We use the same seven industrial circuits from [13] as our test cases. In this experiment, each module has size constraint equal to 60% of the circuit size. Therefore, the size constraint restricts replication to 20% of the total size.

We compare our algorithm TPRG to the Fiduccia-Mattheyses (FM) [3] algorithm, and LAMP [10]. All programs are run on a single-processor SUN SPARC 10 workstation. The results of FM are chosen from the best of 20 runs each.

Table 1 shows the characteristics of the test cases. The fourth and fifth columns stands for the iteration bound $J$ and the path delay bound $B$ which is defined later with the external-loop constraint. Since s2 and s6 do not have feedback loops, the iteration bounds are equal to zero. The sixth and seventh columns list the crossing edge count of FM and LAMP, respectively. As indicated in [2], the intermodule delay $\delta$ can increase to nearly 100% of the clock cycle period. Therefore, we set $\delta$ to be of 60% of $\max(J, B)$, which is calculated before partitioning in this paper.

<table>
<thead>
<tr>
<th>cir.</th>
<th>#reg.</th>
<th>#comb.</th>
<th>J</th>
<th>B</th>
<th>FM cut</th>
<th>LAMP cut</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>342</td>
<td>8280</td>
<td>6373</td>
<td>5447</td>
<td>2860</td>
<td>1314</td>
</tr>
<tr>
<td>s2</td>
<td>472</td>
<td>3378</td>
<td>0</td>
<td>4421</td>
<td>875</td>
<td>847</td>
</tr>
<tr>
<td>s3</td>
<td>521</td>
<td>6252</td>
<td>2527</td>
<td>3238</td>
<td>1422</td>
<td>1629</td>
</tr>
<tr>
<td>s4</td>
<td>380</td>
<td>3850</td>
<td>4922</td>
<td>5545</td>
<td>1045</td>
<td>1032</td>
</tr>
<tr>
<td>s5</td>
<td>545</td>
<td>12172</td>
<td>4241</td>
<td>4876</td>
<td>3465</td>
<td>3478</td>
</tr>
<tr>
<td>s6</td>
<td>357</td>
<td>3026</td>
<td>0</td>
<td>3724</td>
<td>848</td>
<td>817</td>
</tr>
<tr>
<td>s7</td>
<td>607</td>
<td>4990</td>
<td>996</td>
<td>3563</td>
<td>1103</td>
<td>1141</td>
</tr>
</tbody>
</table>

Table 1: Characteristics of test cases.

External-Loop Constraint: A system can interact with external systems. Hence, macroscopically, there possibly exist external feedback loops from the primary outputs to the primary inputs. We call this assumption the external-loop constraint. According to the external-loop constraint, we have to take into account the path delay. Given a path $p$ from the primary input pad to the primary output pad, let $d_p$ and $r_p$ be the sum of delay time and the sum of registers on path $p$. The path delay bound of a circuit is defined by:

$$B = \max \frac{d_p}{r_p} \quad \forall \text{IO-path } p$$

In the following, we present the experimental results with the external-loop constraint. The experimental results without the external-loop constraint appears in [7].

6.1 Experiment with External-Loop Constraint

According to the external-loop constraint, we have to take into account the path delay. Then the dominant delay of a given partitioned circuit is:

$$A = \max (J, B)$$
Table 2 gives our experiments. The timing constraint is the same as that in [10]. The data in the first subcolumn are the value of A derived from above equation after partitioning. The T in the second subcolumn is the clock cycle period of the partitioned circuit after retiming.

The reductions on the cut edge counts are as follows. When compared to the FM, TPRG achieved 1.92 ~72.86% with an average of 22.15%. TPRG achieved 0.61 ~73.76% with an average of 23.79%, compared with LAMP. Especially, TPRG improve the cut edge count by 73.76% for s7, compared with LAMP. Due to s7 with many nodes having large fan-out and small fan-in, the cut edge count is dramatically reduced by replication. When compared to the FM and LAMP, the clock cycle period reduction is as follows. TPRG achieved 22.80 ~34.43% with an average of 27.60% for A and 13.04 ~34.42% with an average of 26.25% for T, compared with FM. Even given the same timing constraint, TPRG still get improvement, compared with LAMP. TPRG achieved 0 ~12.91% with an average of 3.43% for A and 0 ~13.78% with an average of 3.97% for T. The number of iterations of TPRG for these seven test cases are 6, 2, 1, 9, 1, 7 and 1, respectively. The size overheads are 19.96%, 0.95%, 13.82%, 19.66%, 20.00%, 19.93% and 12.48%, respectively.

<table>
<thead>
<tr>
<th>cir.</th>
<th>A</th>
<th>T</th>
<th>A</th>
<th>T</th>
<th>A</th>
<th>T</th>
<th>cut</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>8371</td>
<td>9235</td>
<td>6373</td>
<td>6653</td>
<td>6373</td>
<td>6587</td>
<td>2905</td>
</tr>
<tr>
<td>s2</td>
<td>7074</td>
<td>7215</td>
<td>5200</td>
<td>5310</td>
<td>5080</td>
<td>5130</td>
<td>723</td>
</tr>
<tr>
<td>s3</td>
<td>5338</td>
<td>5444</td>
<td>4019</td>
<td>4099</td>
<td>3500</td>
<td>3534</td>
<td>1018</td>
</tr>
<tr>
<td>s4</td>
<td>8239</td>
<td>8651</td>
<td>6160</td>
<td>7585</td>
<td>6360</td>
<td>7585</td>
<td>905</td>
</tr>
<tr>
<td>s5</td>
<td>7606</td>
<td>8132</td>
<td>6366</td>
<td>6495</td>
<td>5824</td>
<td>5882</td>
<td>2881</td>
</tr>
<tr>
<td>s6</td>
<td>6544</td>
<td>6544</td>
<td>4502</td>
<td>5042</td>
<td>4502</td>
<td>5042</td>
<td>812</td>
</tr>
<tr>
<td>s7</td>
<td>5103</td>
<td>5227</td>
<td>3644</td>
<td>3955</td>
<td>3637</td>
<td>3891</td>
<td>299</td>
</tr>
</tbody>
</table>

Table 2: The experiment with external-loop constraint

6.2 Experiment with Tighter Constraint

By using binary search, we can determine the minimum dominate delay A achieved by our algorithm. Given a ratio K, if our algorithm cannot find a partition with dominate delay A no greater than K within 20 iterations, it fails; otherwise it success and we can go ahead to tighter timing constraints.

Table 3 show the experimental results. #iter. denotes how many iterations TPRG takes to terminate and over denotes the percentage of the size overhead of duplicated nodes. 10.08% improvement on the clock period and 20.65% reduction on the cut edge count are achieved, compared with LAMP approach. Compared with FM approach, TPRG can achieved 33.33% improvement on the clock period and 18.8% reduction on the cut edge counts.

<table>
<thead>
<tr>
<th>cir.</th>
<th>K</th>
<th>cut</th>
<th>A</th>
<th>T</th>
<th>#iter.</th>
<th>over.</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>6373</td>
<td>2805</td>
<td>6373</td>
<td>6587</td>
<td>6</td>
<td>19.96</td>
</tr>
<tr>
<td>s2</td>
<td>4700</td>
<td>785</td>
<td>4569</td>
<td>4614</td>
<td>8</td>
<td>2.74</td>
</tr>
<tr>
<td>s3</td>
<td>3300</td>
<td>1172</td>
<td>3300</td>
<td>3325</td>
<td>13</td>
<td>19.75</td>
</tr>
<tr>
<td>s4</td>
<td>6360</td>
<td>905</td>
<td>6360</td>
<td>7505</td>
<td>9</td>
<td>19.66</td>
</tr>
<tr>
<td>s5</td>
<td>5500</td>
<td>2879</td>
<td>5469</td>
<td>5539</td>
<td>16</td>
<td>19.97</td>
</tr>
<tr>
<td>s6</td>
<td>4300</td>
<td>857</td>
<td>4038</td>
<td>4318</td>
<td>14</td>
<td>19.95</td>
</tr>
<tr>
<td>s7</td>
<td>3565</td>
<td>296</td>
<td>3565</td>
<td>3603</td>
<td>7</td>
<td>16.78</td>
</tr>
</tbody>
</table>

Table 3: Results with tighter constraint

References