Performance-Driven Partitioning Using Retiming and Replication*

*Lung-Tien Liu, Minshine Shih**, Nan-Chi Chou, Chung-Kuan Cheng, and Walter Ku*

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

**Department of EECS
University of California, Berkeley
Berkeley, California 94720

Abstract

We propose a novel paradigm for two-way circuit partitioning which minimizes the clock cycle. The replication technique is suggested for feedback loops to minimize the impacts of intermodule delays and the crossing edges when necessary. A flow timing cut is devised to produce partitions which can be guaranteed to achieve clock cycles equal their lower bound with respect to the partitions using retiming. When the clock cycle optimization is the major objective and feedback loop sizes are not large, we propose an efficient, easy to implement algorithm which still guarantees achieving the lower bound clock cycle with respect to its partition. Experimental results have shown our algorithms can achieve an average of 15% clock cycle time reduction compared to the best retimed results produced by 20 runs on each test case using a Fiduccia-Mattheyses algorithm.

1 Introduction

A synchronous digital system can be represented by a *directed* graph, $G(V = R \cup C, E)$, where R is the set of registers and C is the set of combinational blocks. E is the set of directed edges corresponding to signal flows in the system. A two-way partition P of $G(V, E)$ maps V into two modules, (V_1, V_2), $V_1 \cup V_2 = V$. Due to replication, V_1 and V_2 may overlap. An edge $e = (u, v)$ is a *crossing edge* of P if one node is in V_1 and the other is in V_2. We assume registers and non-crossing edges are of *zero delay*. The intermodule delay δ is a technology dependent constant. Given a feedback loop i, let ℓ_i, $\widehat{d}_i$, and r_i be the sum of combinational block delays, the sum of edge delays, and the number of registers on loop i respectively. The *iteration bound* can be defined as [7]:

$$L = \max\ ((\ell_i + \widehat{d}_i)/r_i) \qquad \forall\ \text{loop} i\ . \quad (1)$$

We make following assumptions in this paper:

1. The intermodule delay is less than the clock period.
2. Data flow are fine-grained in nature.
3. Replicating nodes from other modules is feasible.
4. There exist external feedback loops between the primary I/O. Furthermore, there is at least one extra clock cycle slackness associated with each external feedback.

*This work was supported in part by grants from NSF I/UCR Center, ICAS and NSF MIP-9117328 as well as AT&T, Hughes, and Quickturn under MICRO.

Although replication can be used to reduce the number of crossing edges [3], we use it as an effective approach to avoid extra intermodule delays introduced by cutting feedback loops. This makes our replication objective different from [3], which will be discussed in Section 3.

2 The Timing-Optimal Problem

According to assumption 4, we need to consider the path delay. Let d_j, $\widehat{d}_j$, and r_j be the sum of functional block delays, the sum of intermodule delays, and the number of registers on path j between the primary I/O. The *path delay bound* of a circuit is defined by:

$$D = \max\ (d_j + \widehat{d}_j)/(r_j - 1), \quad (2)$$

$\forall$ path j between the primary I/O. Then the dominant delay T of a given circuit partitioned by P is

$$T = \max\ (L,\ D)\ . \quad (3)$$

Let $M(P)$ denote the minimum cycle time of the circuit partitioned by P, which can be achieved by retiming [4, 5]. We term a partitioning P *bound-optimal* if $T = M(P)$. A partition P is *timing-optimal* if $M(P) \leq M(P')$ for any other partition P'. Now we state **the timing-optimal partitioning problem** as follows:

Given a data flow graph $G(V = R \cup C, E)$ with each node v_i of size(v_i), size constraint S_1 and S_2, and intermodule delay δ, find a timing-optimal partition $P = (V_1, V_2)$ and minimize the number of crossing edges as a secondary objective, subject to $|V_1| \leq S_1$ and $|V_2| \leq S_2$.

Figs. 1 and 2 illustrate the essence of the timing-optimal partitioning problem, where registers are represented by rectangles and are labeled using uppercases. Combinational blocks are represented by circles and are labeled using lowercases. Shaded octagons denote crossing edges. We assume combinational block delays are one unit and intermodule delays δ are two units. Given a circuit in Fig. 1(a), the clock cycle is dominated by the longest combinational delay between registers, which is from A to B with a delay of 3 units. However, according to equation (1), the iteration bound is determined by the left loop, which is equal to $6/3 = 2$. Hence, if we move B to a new location as indicated by the dashed line, the longest path is from A to B or from B to C. Both have a shorter delay of 2 units which equals the iteration bound.

Suppose we partition the circuit into two parts, modules I and II (Fig. 1(b)). The clock cycle is 5 units before retiming because of the delay on the longest path from

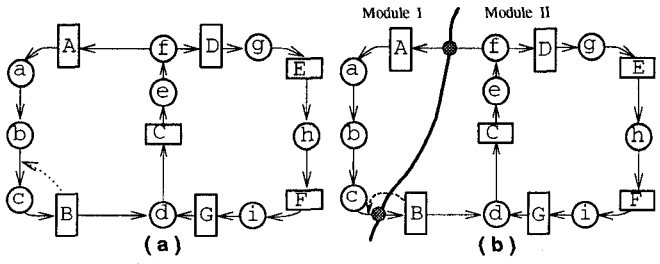

Figure 1: *(a) Retiming reduces the delay from 3 units of original circuit to 2 units. (b) Retiming reduces the delay of partitioned circuit from 5 to 4 units which is not bound-optimal.*

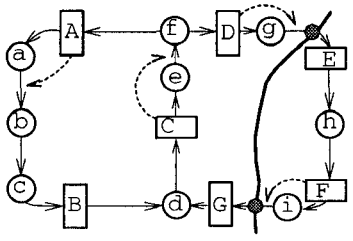

Figure 2: *A bound-optimal and timing-optimal partition with 2 units delay using retiming.*

A to B. Even after retiming which shifts B to its new location as indicated by the dashed line, the delay is more than 3 units. In Fig. 2, before retiming, the clock cycle is 3 units; hence, compared to Fig. 1(b), a better choice of partition can automatically reduce the delay. If we perform the retiming as shown by the dashed lines, the delay in Fig. 2 is reduced to 2 units which is also the iteration bound; hence it is bound-optimal. Furthermore, since it can be observed that 2 units delay is the optimum for the given circuit, Fig. 2 is also a timing-optimal partitioning; hence is preferred.

3 Theoretical Aspects

For brevity, we omit all proofs in this section, and refer them to [6]. Given a data flow graph $G(V = R \cup C, E)$, let $W(u, v)$ denote the minimum number of registers among all paths from combinational nodes u to v. We call a path $p_{u,v}$ from u to v a *critical path* if the number of registers it contained, denoted by $w(p_{u,v})$, equals $W(u, v)$. Let $D(u, v)$ be the maximum total propagation delay among all critical paths $p_{u,v}$. A retiming of a data flow graph $G(V = R \cup C, E)$ is an integer labeling of combinational nodes: $\Pi : C \rightarrow Z$. The retiming specifies a transformation of the original graph in which registers are added and removed so as to change the graph G into a new graph with vertex set $V' = R_\Pi \cup C$. Let $w_\Pi(p_{u,v})$ denote the number of registers of path p after retiming Π. According to [5], we have equation (4), Theorem 1, and Corollary 1:

$$w_\Pi(p_{u,v}) = w(p_{u,v}) + \Pi(v) - \Pi(u). \quad (4)$$

Theorem 1 *Let $G(V = R \cup C, E)$ be a data-flow graph, and K be an arbitrary positive real number. Let Π be a function from C to integers, i.e., $\Pi : C \rightarrow Z$. Then Π is a legal retiming of G, which can achieve a clock cycle time of K iff: (1) $\Pi(u) - \Pi(v) \leq w(p_{u,v})$ for any path p from combinational nodes u to v. (2) $\Pi(u) - \Pi(v) \leq w(p_{u,v}) - 1$ for any two combinational nodes u and v such that $D(u, v) > K$.*

Given a graph $G(V = R \cup C, E)$ and a constant K, a new graph $G'(V' = C, E')$ can be construct as follows. Initially, $E' = \emptyset$. For any two combinational nodes u and v, if there exists at least one path from u to v, we add an edge (u, v) to E'. If $D(u, v) > K$, we associate the edge (u, v) a cost $W(u, v) - 1$. If $D(u, v) \leq K$, edge (u, v) is assigned cost $W(u, v)$. We call these operations the *transitive transformation*. Hence the following Corollary comes in order according to Theorem 1.

Corollary 1 *Given a graph $G(V = R \cup C, E)$ and a constant K, G can be retimed to achieve a clock cycle time of K iff G' does not have loop with negative cost.*

Given a simple loop ℓ, let t_ℓ, d_ℓ, and r_ℓ denote the total delay, the total delay of coarse-grained structures, and the total number of registers in loop ℓ respectively.

Lemma 1 *Given a graph $G(V = R \cup C, E)$ and a constant K, if each simple loop ℓ of G satisfies $(t_\ell + d_\ell)/r_\ell \leq K$, G can be retimed to achieve a cycle time equals K.*

When partitioning a circuit, crossing edges with delays are introduced into the circuit. Since these delays cannot be decomposed, a partitioned circuit contains both fine-grained and coarse-grained nodes. Given a path p, we term p is cut k-times if there are k crossing edges in p. From assumption 4, shifting of primary I/O registers during retiming is allowed.

Theorem 2 *Given a partition P over $G(V = R \cup C, E)$ and an intermodule delay $\hat{d}$, let $T = \max(L, D)$ (equation (3)). If each path between the primary I/O is cut at most once and each loop is not cut, then P can be retimed to achieve a clock cycle time of T.*

In case that a path is cut more than once, a bound-optimal circuit cannot be guaranteed using retiming. More formally, given a partition P over $G(V = R \cup C, E)$ and a intermodule delay δ, let $T = \max(L, D)$ (equation (3)). If any path between the primary I/O is cut more than once, the partition P cannot guarantee to achieve a clock cycle of T by retiming.

To illustrate the problem complexity, we introduce a simplified problem:

The simplified timing-optimal partitioning problem: *Given an acyclic graph $G(V = R \cup C, E)$ with each node v_i has size of size(v_i), size constraint S_1 and S_2, a crossing edge delay δ, and a constant K, find a two-way partition $P = (V_1, V_2)$ subject to $|V_1| \leq S_1$, $|V_2| \leq S_2$, $D \leq K$, and each path between the primary I/O is cut at most once, where D is the path delay bound of P defined in equation (2).*

From [6], we have following theorems:

Theorem 4 *The simplified timing-optimal partitioning problem with unit size combinational blocks and zero size*

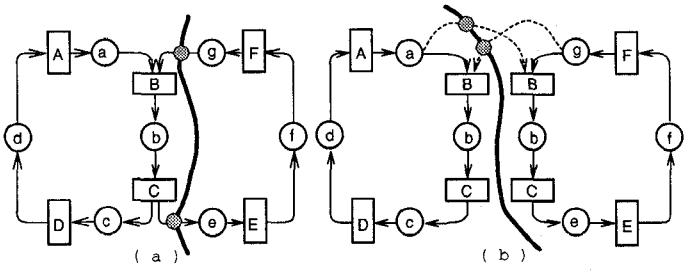

Figure 3: *Reduce iteration bound L through replication. (a) Before replication, $L = 8/4 = 2$. (b) Replicating B, c, and C, $L = 12/8 = 1.5$.*

registers is $\mathcal{NP}$-complete.

Theorem 5 *The timing-optimal partitioning problem stated in Section 2 is $\mathcal{NP}$-complete.*

4 Heuristics

Intuitively, network flow based algorithms should cut paths much fewer times than node-exchange based algorithms such as [2] due to the inject-and-saturate flow nature. Furthermore, the flow-based algorithm is a global-oriented partitioning approach. Hence, we propose an algorithm called the Flow Timing Cut (FTC). By Theorem 2, we merge strongly connected components into supernodes and perform partitioning on the resulted acyclic graph. However, since we may have huge supernodes from the above merging, we apply replication_cut prior to the partitioning; otherwise we proceed without replication. The algorithm is listed as follows:

1. Condense each strongly connected component in $G(V, E)$ into one supernode to obtain an acyclic graph $G'(V', E')$;
2. Let tsize= $\sum_{v \in V'} \text{size}(v)$ and $replicate = 0.2$.
   If ($\exists$ size(supernode) > tsize $\cdot$ *replicate*)
   return(replication_cut(G)); else do (a) to (c):
   (a) Invoke saturate_network (G', Δ, α) to saturate G' with flow to get a distance function d for each node.
   (b) Select a partition P according to d.
   (c) Adjust P such that each path between the primary I/O is cut at most once.

Procedure replication_cut() basically implements the idea shown in Fig. 3, which is aimed to reduce the iteration bound (equation (1)) L; thereby admits higher possibility minimizing the dominant delay T (equation (3)). Assuming delays for registers, combinational blocks, and intermodule δ are zero, one, and two units respectively. Suppose we have a partitioned circuit shown in Fig. 3(a), the corresponding iteration bound L is determined by cycle B, b, C, e, E, f F, g, B. Since the total delay in this cycle is 8 and the number of registers is 4, $L = 2$. However, if we replicate nodes B, c, and C, as shown in Fig. 3(b), the dominant cycle is the whole circuit with $L = 12/8$ which is 25% less than Fig. 3(a).

Given a circuit G', a flow increment Δ, and a constant α, saturate_network() works as follows. Initially, we associate each edge with capacity and distance of 1, and set *slack* of an edge e = *cycle time* minus the delay of e, where cycle time is provided by the designer before timing optimization. The algorithm then randomly pick primary input and output nodes s and t, and find a shortest path from s to t, according to each edge's distance. An increment of flow with amount Δ is injected into the path, and the distance of each edge is updated by an exponential function $d(e) = \exp(\frac{\alpha \cdot f(e)}{slack})$ to penalize congested edges. Saturated edges are removed from G'. This flow injection process is repeated until a two-way partition is obtained. In this paper, we follow [9] to set $\Delta = 0.01$ and $\alpha = 10$.

ckt	# reg	#comb. blocks	L	D	FM cut	TTC cut	FTC cut
s1	342	8280	6373	5447	2860	3144	3043
s2	472	3378	0	4421	875	878	948
s3	521	6325	2527	3238	1422	2236	1952
s4	380	3850	4922	5545	1045	1467	1258
s5	545	12172	4241	4876	3465	4889	4889
s6	357	3026	0	3724	848	1175	1004
s7	607	4990	996	3563	1103	1304	1304

Table 1: *Characteristics of test cases and crossing edges cut by different partitioning algorithms.*

When the clock cycle optimization is the major objective and feedback loop sizes are not large, we propose an efficient and easy to implement algorithm, the *topological timing cut* (TTC). Partitions generated by TTC will still guarantee to cut the paths between the primary I/O at most once. The algorithm is listed below.

1. Same as the step 1 of FTC.
2. Put all nodes of G' into partition I. Move one node at a time, by topological order of G', from partition I to partition II. Record the minimum partition P'.
3. Repeat step 2 with the reverse topological order of G', get P''.
4. Return $min(P', P'')$.

Given a data flow graph $G(V = R \cup C, E)$, let $m = |E|$, and $n = |V|$. The complexities for FTC and TTC are $O(\max((\frac{m}{\Delta}m), n \log n, mn),)$ and $O(m + n \log n)$. The detail derivation is referred to [6].

5 Experimental Results

We use the same seven industrial circuits from [8] as our test cases. Five of these circuits contain feedback loops. However, among these loop-associated circuits, the sizes of all clusters are relatively small—less than one tenth the size of its corresponding circuit. Because of the peculiarity of our test cases, there is no need to perform the replication for these test cases. We compared our algorithms to the Fiduccia-Mattheyses (FM) [2]. The results of FM are chosen from the best of 20 runs each. The left partition of Table 1 shows the characteristics of these test cases. The right partition lists the number of crossing edges cut by various algorithms.

ckt	FM ($\widehat{T}$)			FM (T)			TTC (T)			FTC (T)		
	$\delta = 40$	60	80	$\delta = 40$	60	80	$\delta = 40$	60	80	$\delta = 40$	60	80
s1	8181	9456	10730	6373	6373	6373	6373	6373	6373	6373	6373	6373
				7096	8371	9645	5818	6161	7275	6156	7076	8350
s2	6189	7074	7958	0	0	0	0	0	0	0	0	0
				6189	7074	7958	4421	5207	6091	4556	5342	6226
s3	6101	6801	7501	3441	3908	4375	2527	2527	2527	2527	2527	2527
				4638	5338	6038	4060	4760	5460	4276	4976	5676
s4	8149	9258	10367	4922	4922	4922	4922	4922	4922	4922	4922	4922
				7130	8239	9348	6552	7661	8770	6974	8083	9192
s5	8460	9435	10410	5293	6268	7243	4241	4241	4241	4241	4241	4241
				6691	7666	8641	6699	7674	8649	6699	7674	8649
s6	7716	8656	9596	0	0	0	0	0	0	0	0	0
				5604	6544	7484	4671	5611	6551	4394	5334	6274
s7	5884	6597	7309	996	996	996	996	996	996	996	996	996
				4390	5103	5815	3563	3897	4418	3563	3897	4418

Table 2: *Delay information from different partitioning algorithms when assuming δ = 40, 60, and 80% of T^*.*

Since, as indicated by [1], the intermodule delay is increasing to nearly 100% of the clock cycle, we set δ to be of 40, 60, and 80% of $T^* = \max(L, D)$ which is calculated using equation (3) before partitioning, and used these values to perform experiments for different algorithms. Table 2 gives the detailed information of our experiments. In the first row, $\widehat{T}$ associated with FM is the maximum delay between registers before retiming. T is derived from equation (3) after partitioning. However, for FM, T only serves as a *lower bound* of the delay. This is because the partitions generated by FM are likely to be non-bound-optimal. By contrast, the results produced by our algorithms (FTC and TTC) are guaranteed to be able to achieve a clock cycle of T. For Columns 3 – 5 in Table 2, each test case consists of two row of data. The first and second rows are the iteration bound and the path delay bound respectively. As we mentioned before, there are two test cases which have no feedback loop, *s2* and *s6*. Hence their iteration bounds are listed as "0".

In case that there is no external feedback loop from the primary output to the primary input, L, calculated after partitioning, will dominate the clock cycle time during retiming. However, if $L < \delta$, then δ will dominate the clock cycle because δ is not decomposable. As a result, when compared to the FM without retiming, the clock cycle time can be reduced by an average of 57.38% for all our test cases.

From Assumption 4, when compared to the FM, i.e., without retiming, the clock cycle time reductions are as follows. When δ = 40% of T^*, FTC achieved 14.42 ~ 43.05% with an average of 28.10%. TTC achieved 19.60 ~ 39.46% with an average of 29.03%. When δ = 60% of T^*, FTC achieved 12.69 ~ 40.93% with an average of 26.76%. TTC achieved 17.25 ~ 40.93% with an average of 28.74%. When δ = 80% of T^*, FTC achieved 11.33 ~ 39.55% with an average of 24.47%. TTC achieved 15.41 ~ 39.55% with an average of 26.64%.

When compared to the FM with retiming, If δ = 40% of T^*, FTC achieved -0.12 ~ 26.38% with an average of 12.85%. TTC achieved -0.12 ~ 28.57% with an average of 14.04%. if δ = 60% of T^*, FTC achieved -0.10 ~ 24.48% with an average of 12.95%. TTC achieved -0.10 ~ 26.39% with an average of 15.08%. If δ = 80% of T^*, FTC achieved -0.09 ~ 24.02% with an average of 11.94%. TTC achieved -0.09 ~ 24.57% with an average of 14.31%.

References

[1] D. Doane and P. Franzon *ed., Multichip Module Technologies and Alternatives -The Basics"*, Van Nostrand Reinhold, New York, 1993, pp. 666 - 667

[2] C. Fiduccia and R. Mattheyses, "A Linear Time Heuristic for Improving Network Partitions," *Proc. 19th ACM/IEEE DAC*, 1982, pp. 175 - 181.

[3] J. Hwang and A. Gamal, "Optimal Replication for Min-Cut Partitioning", *Proc. ICCAD*, Nov., 1992, pp. 432 - 435.

[4] C. Leiserson and J. Saxe, "Optimizing Synchronous Systems," *J. VLSI & CS*, Vol. 1, No. 1, 1983, pp. 41 - 67.

[5] C. Leiserson and J. Saxe, "Retiming Synchronous Circuitry," *Algorithmica*, Vol. 6, No. 1, 1991, pp. 5 - 35.

[6] L.-T. Liu, N.-C. Chou, and C.-K. Cheng, "Performance-Driven System Partitioning," *Technical Report CS93-290*, University of California, San Diego, Apr. 1993.

[7] K.Parhi and D.Messerschmitt, "Static Rate-optimal Scheduling of Iterative Data-Flow Programs via Optimum Unfolding," *IEEE T. Computers*, V.40, N.2, 1991, pp. 178-195.

[8] M. Shih and E. Kuh "Quadratic Boolean Programming for Performance-Driven System Partitioning," *Proc. 30th ACM/IEEE DAC*, 1993, pp. 761-765.

[9] C.-W. Yeh, C.-K. Cheng, and T.-T. Lin, "A Probabilistic Multicommodity-Flow Solution to Circuit Clustering Problems," *Proc. ICCAD*, Nov. 1992, pp. 428 - 431.