Data Flow Partitioning for Clock Period and Latency Minimization *

*Lung-Tien Liu, Minshine Shih** and Chung-Kuan Cheng*

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 an efficient performance-driven two-way partitioning algorithm to take into account clock cycle period and latency with retiming. We model the problem with a Quadratic Programming formulation to minimize the crossing edge count with nonlinear timing constraints. By using *Lagrangian Approach on Modular Partitioning (LAMP)*, we merge nonlinear constraints to the objective function. The problem is then decomposed into primal and dual two subprograms. The primal and dual problems are solved by a Quadratic Boolean Programming approach and by a subgradient method using cycle mean method, respectively. Experimental results show our algorithm achieves an average of 23.25% clock cycle period and 19.54% latency reductions compared to the Fiduccia-Mattheyses algorithm. In terms of the average number of the crossing edges, our results are only 1.85% more.

1 Problem Formulation

A synchronous digital system can be represented by a *directed* graph, $G(V = R \cup C,\ E)$, where R is the set of register nodes and C is the set of combinational block nodes. Each node i has an associated size s_i and delay d_i. E is the set of directed edges which correspond to signal flow in the system. Each edge (i, j) is associated with an attribute $c_{i,j}$, which denotes the number of the interconnections from nodes i to j.

A two-way partition $P = (V_1, V_2)$ maps V into two modules, such that $V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$. The capacity limits of these modules are denoted by S_1 and S_2, respectively. An edge (i, j) is a *crossing edge* of P if node i and node j are in different subsets V_1 and V_2. We assume register nodes and non-crossing edges are of *zero delay*. The crossing edges have an intermodule delay δ determined by technologies.

1.1 Assumptions

We make the following assumptions in this paper:

1. The intermodule delay is less than the desired clock period, T.
2. Data flow are fine-grained in nature.

Although we assume the combinational blocks are fine-grained, some structures, e.g. the delays on crossing edges, are inherently coarse-grained and cannot be split.

*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.

1.2 Iteration Bound and Latency

Iteration Bound: Suppose the data flow contains no loop or feedback, we can utilize the concept of parallel and pipeline processing to increase the throughput arbitrarily. However, this is not the case for a data flow with feedbacks. Feedbacks impose an inherent *lower bound* on the loop iteration [11]. Given a feedback loop l, let d_l, $\widehat{d_l}$, and r_l be the sum of combinational block delays, the sum of edge delays, and the number of registers on loop l respectively. The delay-to-register ratio of l is equal to $\frac{d_l + \widehat{d_l}}{r_l}$. The *iteration bound* is defined by:

$$J = \max\ (\frac{d_l + \widehat{d_l}}{r_l}) \qquad \forall \text{ loop } l\ . \tag{1}$$

The iteration bound of a given circuit stands for the lower bound of the clock period achieved by retiming.

Cycle Mean Problem: For the special case that each edge contains one register, the iteration bound becomes a cycle mean problem [5]. In our test case, each combinational block node connects between two register node. Thus, we can transform the iteration bound into the cycle mean problem. Karp [5] proposes an $O(nm)$ algorithm to solve the cycle mean problem.

Latency: Given a path p, we use $r(p)$ to denote the number of register on the path. Let $W(i, j)$ denote the minimum $r(p)$ among all possible path from i to j, i.e. $W(i, j) = \min\ r(p)\ \forall\ p$ from i to j. Let T be the clock period. We define the latency between primary input i and primary output j to be the minimum number of clock periods that pass between the signal arrival at node i and its first effect on the signal at node j, i.e. $(W(i, j) - 1) * T$. Note that the primary inputs and outputs are register nodes. The signal at primary input i takes $W(i, j) - 1$ clock periods to first arrive at primary output j. However, if there is no path between i and j, we set its latency to zero. Thus, we define the latency of the whole system be the maximum latency among all possible input output pairs, [7, 4, 12] i.e.

$$N = \max\ (W(i, j) - 1) * T \tag{2}$$

where i and j are primary input and primary output, repectively.

Latency Bound: We define a path p from i to j a *critical path* if the number $r(p)$ of registers it contained equals $W(i, j)$. If node i is the primary input and node j is the primary output, the path p is also called the *IO-critical*

path. Let d_p and $\widehat{d_p}$ be the sum of functional block delays and the sum of intermodule delays on path p. The input signal at node i will take at least $d_p + \widehat{d_p}$ delays to arrive at node j. Let $D(i,j) = max\ d_p + \widehat{d_p}$ for all critical path p. Leiserson *et al.* [8] propose an all-pairs shortest-paths algorithm to calculate the quantities W and D. Since the critical paths from nodes i to j determine the latency between i and j, the *latency bound* is defined as follows:

$$M = \max\ d_p + \widehat{d_p} \qquad \forall \text{ IO-critical path } p. \quad (3)$$

Leiserson's algorithm can be adopted to identify the latency bound and the edges that contribute to the bound. The latency bound also imposes the lower bound on the latency of the circuit. We want to generate a partition with small iteration and latency bounds.

Retiming: Given data flow graph $G(V = R \cup C,\ E)$, let $I_R \subseteq R$ be the set of primary inputs; $O_R \subseteq R$ be the set of primary outputs. A retiming of a data flow graph $G(V = R \cup C, E)$ is an integer labeling of combinational, primary input, and primary output nodes: $\Pi : C \cup I_R \cup O_R \rightarrow Z$. The retiming specifies a transformation of the original graph in which registers, except primary inputs and primary outputs, are added or removed so as to change the graph G into a new graph $G_\Pi = (V_\Pi = R_\Pi \cup C, E_\Pi)$. Let $r_\Pi(p_{u,v})$ denote the number of registers of path p after retiming Π. According to [8], we have equation (4).

$$r_\Pi(p) = r(p) + \Pi(j) - \Pi(i). \quad (4)$$

From (4), we know if a path p from nodes i to j is critical before retiming, p is still critical after retiming. We derive the following theorem:

Theorem 1 *Let $G(V = R \cup C, E)$ be a data-flow graph, K be an arbitrary positive real number, and H be an integer. Let Π be a function from $C \cup I_R \cup O_R$ to integers, where $I_R \subseteq R$ and $O_R \subseteq R$. Then Π is a legal retiming of G, which can achieve a clock period of K and a latency of H clock periods iff:*

(i) $\Pi(i) - \Pi(j) \leq W(i,j)$ *for any two nodes i and j.*

(ii) $\Pi(i) - \Pi(j) \leq W(i,j) - 1$ *for any two nodes i and j such that $D(i,j) > K$.*

(iii) $\Pi(j) - \Pi(i) \leq H - W(i,j)$ *for any $i \in I_R$ and any $j \in O_R$.*

From Theorem 1, the problem to find out a retiming which can achieve a clock period of K and latency of H clock periods is equal to one to find out a function Π from $C \cup I_R \cup O_R$ to integers which satisfies all the constraints in (i), (ii), and (iii) of Theorem 1. In other words, we need to determine the feasible values for all the unknowns $\Pi(i)$ under a set of inequality constraints with the form of $\Pi(i) - \Pi(j) \leq u_{i,j}$, where $u_{i,j}$ is a constant. Constraints systems with such format arise in shortest paths problems. We can use the Bellman-Ford algorithm [6] to solve it.

1.3 The Performance-Driven Partitioning Problem

Since the performance of the circuit is measured by both clock cycle period and latency, we want to generate the partition which can achieve a good clock cycle period and latency with retiming.

Given a partition $P = (V_1, V_2)$, let $X(P)$ and $Y(P)$ denote the minimum cycle period and latency which can be obtained by retiming the circuit partitioned by P. Now we state **the performance-driven partitioning problem** as follows:

Given a data flow graph $G(V = R \cup C,\ E)$ with each node i of size s_i, two numbers K and H, size constraints S_1 and S_2, and intermodule delay δ, find a partition $P = (V_1, V_2)$ with the minimum number of crossing edges, subject to $\sum_{i \in V_1} s_i \leq S_1$, $\sum_{i \in V_2} s_i \leq S_2$, $X(P) \leq K$, and $Y(P) \leq H$.

Let $J(P)$ denote the iteration bound and $M(P)$ denote the latency bound with respect to partition P. In practice to simplify the performance-driven partitioning problem, we will replace the cycle period $X(P)$ and latency $Y(P)$ with their lower bounds $J(P)$ and $M(P)$, respectively. Retiming is then performed after the partitioning to derive the exact system cycle period and latency.

We use examples shown in Fig. 1 , 2 , 3 and 4 to illustrate the essence of the performance-driven partitioning problem. In Fig. 1 , 2 , 3 and 4, shaded octagons denote crossing edges. In these examples, combinational block delays are one unit and intermodule delays δ are two units. Register I and O denote primary input and output, respectively.

Given a circuit in Fig. 1, the clock cycle period is dominated by the longest combinational node delay between registers, which is from A to B with a delay of 3 units. There are two paths from nodes I to O. One path has 9 registers, the other has 10 registers. So, the latency of this circuit is 8 clock periods, i.e. 24 units. However, using retiming, we can 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. Furthermore, the latency become 16 units, which is much less than the original one, 24 units. Because the iteration bound, which is determined by the left loop, is 2 units, we cannot obtain a smaller clock period.

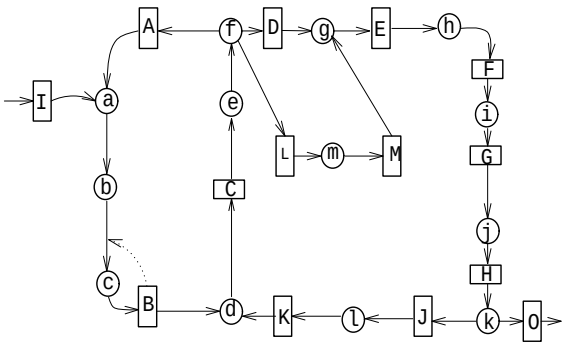

Figure 1: Final clock period and latency are 2 and 16 units

Suppose the circuit is partitioned into two modules (Fig. 2). The clock cycle period is 5 units. Even after retiming which shifts B to its new location as indicated by the dashed line, the delay, 4 units, is more than 3 units. The latency increases to 32 units. In Fig. 3, before retiming, the clock cycle period is 3 units; hence, compared

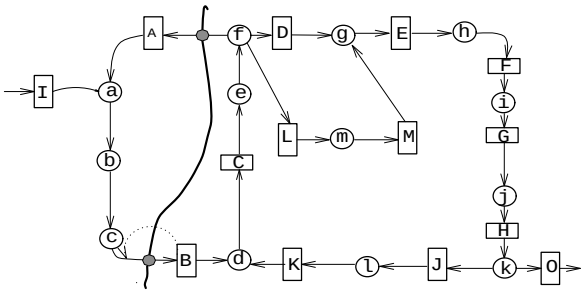

Figure 2: A partition with 4 units delay and 32 units latency

to Fig. 2, 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. 3 is reduced to 2 units. The retiming adds one extra register P between the combinational node k and register O. This turns the latency into 9 clock periods, 18 units. But in Fig. 4, we can achieve a clock period of 2 units. The latency is still 8 clock periods. Hence, Fig. 4 is preferred.

2 Quadratic Boolean Programming Formulation

The performance-driven partitioning problem can be represented by a Quadratic Boolean Programming formulation with nonlinear constraints. We then absorb the nonlinear constraints into the objective function as a Lagrangian. Finally, the Lagrangian is decomposed into primal and dual two subproblems.

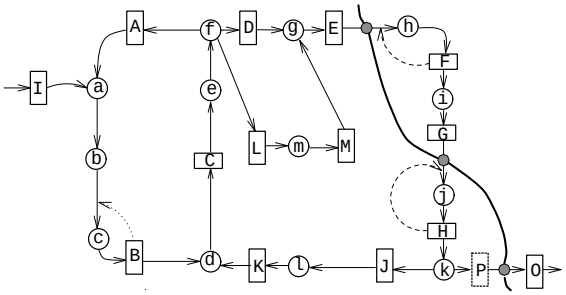

Figure 3: A partition with 2 units delay and 18 units latency

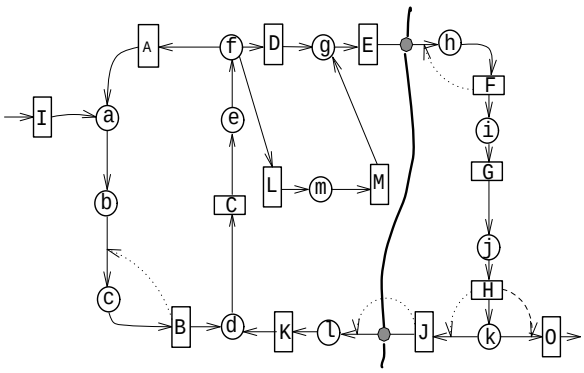

Figure 4: A partition with 2 units delay and 16 units latency

Given the circuit $G(V = R \cup C,\ E)$ with $|V| = n$. A two-way partition can be described by a vector $x = (x_{1,1}, \ldots, x_{1,n}, x_{2,1}, \ldots, x_{2,n})$, where $x_{b,i}$ is 1 if node i is assigned to module b, otherwise $x_{b,i}$ is 0. If nodes i and j are in different modules, the value of the term $x_{1,i}x_{2,j} + x_{2,i}x_{1,j}$ is equal to 1. This contributes one intermodule delay δ into the delay of the edge (i,j). Let $g_l(x)$ and (i,j) denote the delay to register ratio of loop l and the edge from nodes i to j, respectively. $g_l(x)$ can be written as the following formula:

$$g_l(x) \ = \ \frac{d_\ell \ + \ \sum_{(i,j)\in l} \delta(x_{1,i}x_{2,j} + x_{2,i}x_{1,j})}{r_l} \qquad (5)$$

Given a path p, the total delays $h_p(x)$ of p is following:

$$h_p(x) \ = \ d_p \ + \ \sum_{(i,j)\in p} \delta(x_{1,i}x_{2,j} + x_{2,i}x_{1,j}) \qquad (6)$$

Let us formulate the problem. We use an objective function of crossing edge count:

$$\min \sum_{(i,j)\in E} c_{i,j}(x_{1,i}x_{2,j} \ + \ x_{2,i}x_{1,j}) \qquad (7)$$

Subject to the following constraints:

$C1$: (Capacity Constraints)

$$\sum_{i=1}^{n} x_{b,i}s_i \le S_b \qquad \forall\ module\ b \in \{1,2\}\ . \qquad (8)$$

$C2$: (Generalized Upper Bound Constraints)

$$\sum_{b=1}^{2} x_{b,i} = 1 \qquad \forall\ node\ i \in V\ . \qquad (9)$$

$C3$: (Iteration Bound Constraints)

$$g_l(x) \le K \qquad \forall\ \text{loop}\ l\ . \qquad (10)$$

$C4$: (Latency Bound Constraints)

$$h_p(x) \le H \qquad \forall\ \text{IO-critical path}\ p. \qquad (11)$$

Actually, we don't need to consider all loops in $C3$. Because all loops are composed of simple loops, we have the following lemma:

Lemma 1 *Given a number K, if $g_l(x)$ is less than or equal to K for any simple loop l, then $g_l(x)$ is less than or equal to K for all loops l.*

Let π_c and π_p represent the number of the simple loops and IO-critical paths, respectively. Let λ denote the vector $(\lambda_{g_1}, \ldots, \lambda_{g_{\pi_c}}, \lambda_{h_1}, \ldots, \lambda_{h_{\pi_p}})$. Using Lagrangian Relaxation [13], we dualize the constraints (10) and (11) into the objective function (7). The Lagrangian-relaxed problem is as follows.

$$\max_{\lambda \ge 0} \ \min_{x} \ L(x,\lambda) \qquad (12)$$

subject to constraints $C1$ and $C2$, where

$$L(x,\lambda) = \sum_{(i,j)\in E} c_{i,j}(x_{1,i}x_{2,j} + x_{2,i}x_{1,j}) + \sum_{\forall \text{ simple loop } l} \lambda_{g_l}(g_l(x)-K) + \sum_{\forall \text{ IO-critical path } p} \lambda_{h_p}(h_p(x)-H) \quad (13)$$

The Dual Problem: Given vector x, we can represent (13) as a function of variable λ, i.e. $L_x(\lambda)$. Thus, the dual problem can be written as:

$$\max_{\lambda \geq 0} L_x(\lambda) \quad (14)$$

The Primal Problem: Let $F_{i,j}$ and $Q_{i,j}$ denote the set of the simple loops and IO-critical paths passing the edge (i,j). We try to rewrite equation (13) in terms of edges. Let us define $a_{i,j}$ as follows:

$$a_{i,j} = c_{i,j} + \sum_{l\in F_{i,j}} \frac{\delta}{r_l}\lambda_{g_l} + \sum_{p\in Q_{i,j}} \delta\lambda_{h_p} \quad (15)$$

Given vector λ, we can represent (13) as a function of vector x, i.e. $L_\lambda(x)$. Thus, the primal problem can be rewritten as:

$$\min\ L_\lambda(x) = \min \sum_{(i,j)\in E} a_{i,j}(x_{1,i}x_{2,j} + x_{2,i}x_{1,j}) + \beta \quad (16)$$

subject to constraints $C1$ and $C2$, where β represents the constant contributed by λ, the number of registers r_l, the node delay d_l of loop l, etc.

From the above formula, primal problem (16) can be viewed as a Quadratic Boolean objective function with linear constraints. Thus, we utilize a heuristic Quadratic Boolean Programming method (QBP) [1] to calculate the primal problem. Dual problem (14) has the nice properties of continuity and concavity for a hill-climbing algorithm.

3 Subgradient Method on Dual Problem Using Cycle Mean Method

We will use primal-dual iterations to solve the Lagrangian. Once a solution λ of the dual problem (14) is generated, formula (15) is applied to update the edge costs. Given an edge (i,j), we need all the λ_{g_l} and λ_{h_p} values of the simple loops and IO-critical paths passing (i,j) to calculate the equation (15). The number of these simple loops and IO-critical paths can be an exponential function of n, the number of the nodes.

For an optimal solution of problem (14), the Lagrange multiplier λ has the properties that λ_{g_l} and λ_{h_p} are larger than zero only if $g_l(x)$ and $h_p(x)$ are not less than K and H. We define these loops and paths as active loops and paths.

Active Loops and Paths: A simple loop l is called active, if $g_l(x)$ is not less than K. A IO-critical path p is called active, if $h_p(x)$ is not less than H. If a loop or path is not active, we call it inactive.

Active Edges: An edge e is active, if it is covered by an active loop or an active path. If e is not active, we call it inactive.

Dominant Loops and Paths: A simple loop l is called the dominant loop of edge e, if l has the maximum delay-to-register ratio among all loops passing e. We define p to be a dominant path of e, if the total delay of path p is the maximum among all paths passing e.

Dominant Prices: If l is the dominant loop of e, we define the dominant-loop price of e as the delay-to-register ratio of l. If p is the dominant path of e, the total delay of p is defined as the dominant-path price of e.

Given an edge $e = (i,j)$, let f_e be the dominant-loop price of e. r_e denotes the number of registers on the dominant loop of e. Note that we randomly choose one dominant loop to calculate r_e, if e has more than one dominant loops. q_e represents the dominant-path price of e.

We don't generate all the loops and paths. Instead, we determine only dominant prices of each edge. We utilize the minimum cycle mean algorithm [5] to calculate the dominant-loop price of each edge. Leiserson's algorithm [8] to compute the quantities W and D is adopted to compute the dominant-path price of each edge. The edge cost is updated by the subgradient approach.

At the k-th iteration, let x^k and Cut^k denote the derived x and cut of the primal problem, respectively. Let $a^k_{i,j}$ be the cost of edge e at the k-th iteration. We adopt the subgradient method [13] to generate the new edge cost, $a^{k+1}_{i,j}$.

$$a^{k+1}_{i,j} = \max\{0,\ a^k_{i,j} + t^k*(f_e-K)*\frac{\delta}{r_e} + t^k*(q_e-H)*\delta\} \quad (17)$$

t^k is defined as follows:

$$t^k = \frac{\alpha\left|Cut^* - Cut^k\right|}{\sum_{e\in E}|f_e-K|^2 + \sum_{e\in E}|q_e-H|^2} \quad (18)$$

where α and Cut^* are two given positive numbers. By (17), we increase the costs of active edges and decrease ones of inactive edges, using subgradient approach. This captures the actual direction of the edge cost changes of active and inactive edges with respect to the equation (15).

4 Lagrangian Approach on Modular Partitioning (LAMP)

We adopt a Lagrangian Approach on Modular Partitioning (LAMP) which solves the partitioning problem through primal and dual iterations on the Lagrangian. A Quadratic Boolean Programming, QBP, [1] is used to solve the primal problem and generate a solution x. For the dual problem based on x, we call a minimum cycle mean algorithm [5] and Leiserson's algorithm [8] for calculating the quantities W and D to obtain the dominant prices of each edge. We then calculate the subgradient on the dominant prices and update the constants $a_{i,j}$ for

the next primal-dual iteration. The iteration proceeds until the bound of all loops and paths are within the given limits.

In the following algorithm, we initialize a set of parameters K, H and Cut^* from the results of Topological Timing Cut [9], which is very fast and can generate a partition with a good iteration bound. The value of α is set to 1.3 in our experiment.

LAMP Algorithm

1. Assign values to K, H, α and Cut^*.
2. Initialize $k \leftarrow 0$; $a^0_{i,j} = c_{i,j}$.
3. Call QBP [1] to generate a partition $P^k = (V^k_1, V^k_2)$ with crossing edge count $Cut(P^k)$.
4. a. Use the minimum cycle mean algorithm to calculate the iteration of the partitioned circuit and f_e and r_e of each edge e;
   b. Use Leiserson's algorithm to calculate the latency bound and q_e of each edge e;
   c. If the iteration and latency bounds are not greater than K and H respectively, then stop.
5. Compute t^k by equation (18).
6. For each edge $e = (i, j) \in E$:
   Compute $a^{k+1}_{i,j}$ by (17).
7. Set $k \leftarrow k + 1$ and goto 3.

Let $n = |V|$ and $m = |E|$. The time complexity of Leiserson's algorithm is $O(n^3)$, since it is an all-pairs shortest-paths algorithm. The minimum cycle mean algorithm takes $O(nm)$ time. The complexity of QBP [15] is $O(n^2)$.

5 Experimental Results

We use the same seven industrial circuits from [15] as our test cases. All combinational blocks are of unit size except some have size 2 in test case *s1*. Five of these circuits contain feedback loops.

We compare our algorithm to the Fiduccia-Mattheyses (FM) [3] algorithm, and Flow Timing Cut (FTC)[9]. Because TTC and FTC have close results in terms of the number of crossing edges and iteration bound, we don't show TTC's results in this paper. The comparison result between TTC and LAMP can be found in [10]. All algorithms are implemented on a single-processor SUN SPARC 10 workstation under the C/UNIX environment. The results of FM are chosen from the best of 20 runs each. The left partition of Table 1 (columns 2 – 5) shows the characteristics of these test cases. The fifth column stands for the *path delay bound*. A path p is called IO-path, if p is from primary input to output. Given a path p from the primary input to the primary output, let d_p, $\widehat{d_p}$, and r_p be the sum of functional block delays, the sum of intermodule delays, and the sum of registers on path p. The *path delay bound* of a circuit is defined by:

$$B = \max \frac{d_p + \widehat{d_p}}{r_p} \qquad \forall \text{ IO-path } p \qquad (19)$$

ckt	#reg	#comb	J	B	FM	FTC	LAMP
s1	342	8280	6373	5447	2860	3043	3134
s2	472	3378	0	4421	875	948	847
s3	521	6325	2527	3238	1422	1952	1629
s4	380	3850	4922	5545	1045	1258	1032
s5	545	12172	4241	4876	3465	4889	3478
s6	357	3026	0	3724	848	1004	817
s7	607	4990	996	3563	1103	1304	1141

Table 1: Characteristics of test cases

The right partition of Table 1 (columns 6 – 9) lists the number of crossing edges cut by different algorithms.

The reductions of the crossing edge counts are as follows. When compared to the FM, LAMP achieved -14.55 $\sim$ 3.65% with an average of -1.85%. When compared to the FTC, LAMP achieved -2.99 $\sim$ 28.86% with an average of 14.59%.

ckt	FM ($\widehat{T}$)	FM		FTC		LAMP	
		J	T	J	T	J	T
s1	9456	6373	6373	6373	6373	6373	6373
s2	7074	0	2652	0	2652	0	2652
S3	6801	3908	4103	2527	2527	2527	2527
s4	9258	4922	4922	4922	4922	4922	4922
s5	9435	6268	6894	4241	4241	4241	4241
s6	8656	0	2236	0	2236	0	2236
s7	6597	2137	2137	2137	2137	2137	2137

Table 2: Iteration bound J and cycle period T

For performance-driven partitioning, the value of δ should be dertermined. Since, as indicated by [2], the intermodule delay increases to nearly 100% of the clock cycle period, we set δ to be of 60% of $T^* = \max(J, B)$ which is calculated using equation (20) before partitioning, and used these values to perform experiments for different algorithms. Notice that the T^* is obtained before partitioning.

Table 2 gives the detailed information of our experiments for the clock period. In the first row, the $\widehat{T}$ associated with FM is the maximum delay between registers before retiming. In Table 2, except the first and second columns, each column contains two subcolumns. The data in the first subcolumn represents the J derived from equation (1) after partitioning. The T in the second subcolumn is the clock cycle period of the partitioned circuit achieved by retiming. J, calculated after partitioning, will dominate the optimal clock cycle period during retiming. However, if $J < \delta$, then δ will dominate the clock cycle period because δ is not decomposable. Because the size of these loops are quite small and strongly connected, the loops are not cut by all algorithms for most test cases. LAMP, and FTC obtain the same iteration bound and clock period for all test cases. Compared with FM, LAMP achieves 38.41% and 38.48% clock period reductions for $s3$ and $s5$, respectively.

Table 3 gives the detailed information of our experiments for latency. In Table 3, except the first column,

ckt	FM		FTC		LAMP	
	M	M'	M	M'	M	M'
s1	20418	25492	18195	25492	18195	19119
s2	11785	15912	9789	10608	8127	10608
s3	16897	20515	14872	17689	13308	17689
s4	25951	34454	25843	34454	25594	29532
s5	37853	41364	32347	29687	28435	29687
s6	11147	13416	9973	13416	9502	13416
s7	13283	19233	10673	12822	10140	12822

Table 3: Latency bound M and latency M'

each column contains two subcolumns. The data in the first subcolumn represents the M derived from equation (3) after partitioning. The M' in the second subcolumn is the latency of the partitioned circuit obtained through retiming. When compared to the FM and FTC, the latency reductions are as follows. LAMP achieved $1.37 \sim 31.03\%$ with an average of 18.25% for M and $0 \sim 33.33\%$ with an average of 19.54% for M', compared with FM. When compared with FTC, LAMP achieved $0 \sim 16.97\%$ with an average of 7.17% for M and $0 \sim 25.00\%$ with an average of 5.61% for M'.

Digital systems can interact with each others. Even though a system has no feedback loop from its primary output to its primary input, it 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 assumption. According to the external-loop assumption, we have to take into account the path delay. Then the dominant delay of a given partitioned circuit is

$$A = \max(J, B) . \tag{20}$$

If the external-loop assumption holds, Table 4 gives the detailed information of our experiments. The data in the first subcolumn represents the A derived from equation (20) after partitioning. The T in the second subcolumn is the clock cycle period of the partitioned circuit after retiming. LAMP achieved $16.95 \sim 31.20\%$ with an average of 24.92% for A and $13.04 \sim 27.98\%$ with an average of 23.25% for T, compared with FM. When compared with FTC, LAMP achieved $2.54 \sim 21.31\%$ with an average of 12.25% for A and $-0.97 \sim 17.62\%$ with an average of 7.70% for T.

The execution time of LAMP is determined by one of the all-pairs shortest-paths algorithm and the number of the primal and dual iterations. According to our experiments, LAMP stops within 20 iterations for all test cases. The execution times of the seven test cases are ranged from 47 seconds to 329 seconds.

6 Concluding Remarks

We propose an efficient performance-driven two-way partitioning algorithm to take into account both clock cycle period and latency of the partitioned system. Furthermore, we can easily expand our algorithm to K-way partitioning, since QBP can handle K-way partitioning.

ckt	FM ($\widehat{T}$)	FM		FTC		LAMP	
		A	T	A	T	A	T
s1	9456	8371	9238	7076	7076	6373	6653
s2	7074	7074	7215	5342	5342	5206	5310
s3	6801	5338	5444	4976	4976	4019	4099
s4	9258	8239	8631	8083	8083	6360	7505
s5	9435	7666	8432	7674	7827	6366	6493
s6	8656	6544	6544	5334	5429	4502	5042
s7	6597	5103	5227	3897	3897	3644	3935

Table 4: Iteration bound A and cycle period T with external-loop

References

[1] R.E. Burkard and T. Bonniger, "A Heuristic for Quadratic Boolean Programs with Applications to Quadratic Assignment Problems," *European Journal of Operational Research,* 1983, 13, pp. 372 - 386.

[2] Daryl A. Doane and Paul D. Franzon *ed., Multichip Module Technologies and Alternatives –The Basics"*, Van Nostrand Reinhold, New York, 1993, pp. 666 - 667

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

[4] F. Berman, *Personal communication*, 1993.

[5] R. M. Karp, "A characterization of the minimum cycle mean in a digraph", *Discrete Mathematics*, 23, 1978, pp. 309 - 311.

[6] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

[7] E. A. Lee, *Personal communication*, 1993.

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

[9] L.T. Liu, M. Shih, N.C. Chou, C.K. Cheng, and W. Ku, "Performance-Driven Partitioning Using Retiming and Replication," *Proc. IEEE International Conference on Computer-Aided Design,* Santa Clara, Nov. 1993, pp. 269-299.

[10] L. T. Liu and C. K. Cheng, "Data Flow Partitioning for Clock Period and Latency Minimization," *Technical Report CS93-327, University of California, San Diego,* Oct. 1993.

[11] K. K. Parhi, D. G. Messerschmitt, "Static Rate-Optimal Scheduling of Iterative Data-Flow Programs via Optimum Unfolding," *IEEE Trans. on Computers,* Vol. 40, No. 2, 1991, pp. 178-195.

[12] K. K. Parhi, *Personal communication*, 1993.

[13] J. F. Shapiro, *Mathematical Programming: Structures and Algorithms*, Wiley, New York, 1979.

[14] M. Shih, E. S. Kuh, and R.-S. Tsay, "Performance-Driven System Partitioning on Multi-Chip Modules," *Proc. 29th ACM/IEEE Design Automation Conf.,* 1992, pp. 53 - 56.

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