[13] W. Kunz, D. Stoffel, and P. R. Menon, "Logic optimization and equivalence checking by implication analysis," *IEEE Trans. Computer-Aided Design*, vol. 16, pp. 266–281, Mar. 1997.

[14] W. Kunz and D. Pradhan, "Recursive learning: a new implication technique for efficient solutions to CAD problems: test, verification and optimization," *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 1143–1158, Sept. 1994.

[15] K. McElvain.. *LGSynth93 Benchmark Set: Version 4.0* [Online] Available: http://zodiac.cbl.ncsu.edu/CBL_Docs/lgs93.html

[16] J. P. Roth, "Diagnosis of automata failures: a calculus and a method," *IBM J. Res. Develop.*, vol. 10, pp. 278–291, 1966.

Temporal Logic Replication for Dynamically Reconfigurable FPGA Partitioning

Wai-Kei Mak and Evangeline F. Y. Young

***Abstract*—In this paper, we propose the idea of temporal logic replication in dynamically reconfigurable field-programmable gate array partitioning to reduce the communication cost. We show that this is a very effective means to reduce the communication cost by taking advantage of the slack logic capacity available. Given a K-stage temporal partition, the min-area min-cut replication problem is defined and we present an optimal algorithm to solve it. We also present a flow-based replication heuristic which is applicable when there is a tight area bound that limits the amount of possible replication. In addition, we show a correct network flow model for partitioning sequential circuits temporally and propose a new hierarchical flow-based performance-driven partitioner for computing initial partitions without replication.**

***Index Terms*—Dynamically reconfigurable FPGAs, field-programmable gate arrays, logic replication, reconfigurable computing, temporal partitioning.**

I. Introduction

Dynamically reconfigurable field-programmable gate array (DRFPGA) is an important research topic for reconfigurable computing because it has the potential to dramatically improve the logic density by time-sharing logic. Several such devices have been proposed over the years which include [2], [3], [8], [9], [12], [17], and [20]. However, they require an application to be partitioned temporally subject to some new conditions not present in the traditional spatial partitioning problem. In this paper, we address the temporal partitioning problem for DRFPGA with temporal logic replication for communication cost reduction.

DRFPGAs with multiple on-chip configurations allow dynamic reuse of the logic blocks and wire segments by employing more than one on-chip SRAM/DRAM bit to control them (Fig. 1). The time it takes to reconfigure the logic blocks and wire segments of the entire FPGA from the on-chip SRAM/DRAM is on the order of nanoseconds instead of milliseconds as required by reconfiguration from off-chip. This fast on-chip reconfiguration time is critical for throughput driven applications. DRFPGAs have the advantage of much higher logic density over conventional FPGAs. For example, an eight-context DRFPGA fabricated by NEC [9] can accommodate eight times more logic with only a 35% increase in area due to multiplexing. In addition, the prototype DRFPGA has been evaluated in various media/communication applications [9], [10], [18], [23] for reconfigurable processing of some numerically intensive algorithms, and showed both performance and energy consumption improvement over conventional microprocessors by more than an order of magnitude.

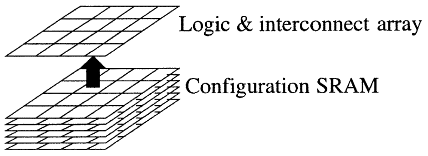


Fig. 1. Conceptual model of a DRFPGA.

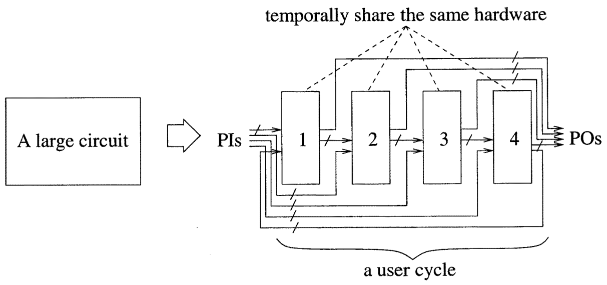


Fig. 2. Partitioning a circuit temporally into four stages.

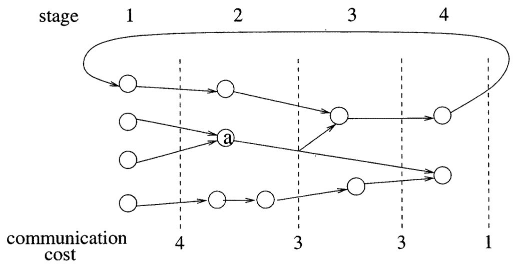


Fig. 3. Communication cost.

To implement an application on a DRFPGA, it has to be partitioned into multiple stages. The configuration of the DRFPGA will be switched continuously to implement each stage one by one in order to perform the functions of the original circuit. Fig. 2 shows a circuit partitioned into stages 1 to 4, the execution sequence will be 1,2,3,4,1,2,3,4, ... To ensure that all computations will be performed correctly when the circuit is divided into stages, certain temporal constraints must be satisfied. For example, to partition a combinational circuit for implementation on a DRFPGA, each logic node must be assigned to a stage no later than any of the nodes that receive input from it to ensure the correctness of the computation of those nodes.

In temporal partitioning, each signal generated in a stage must be buffered until the stage it is last needed. We define the communication cost at a stage as the number of signals that need to be buffered at the end of that stage. An example is shown in Fig. 3. The output of node a has to be buffered at the end of stage 2 and has to remain buffered until stage 4. It is known that the storage needed for buffering signals creates a considerable overhead [4]. Hence, an objective in temporal partitioning is to minimize the communication cost.

Manuscript received May 30, 2002; revised September 3, 2002 and January 17, 2003. This paper was recommended by Associate Editor C. J. Alpert.

W.-K. Mak is with the Department of Computer Science and Engineering, University of South Florida, Tampa, FL 33620-5399 (e-mail: wkmak@csee.usf.edu).

E. F. Y. Young is with the Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, N.T., Hong Kong (e-mail: fyyoung@cse.cuhk.edu.hk).

Digital Object Identifier 10.1109/TCAD.2003.814237

Since there is almost always some slack logic capacity in a stage, we consider how to take advantage of it to optimize the communication cost. In spatial partitioning, it is known that logic replication can be performed to reduce the number of interconnections between components [11], [13], [14], [25]. However, replicating logic temporally has never been suggested or investigated before. In this paper, we propose to use temporal logic replication to effectively exploit the slack logic capacity of a stage to reduce the communication cost. We note that though this paper focuses on temporal logic replication for DRFPGA, the idea can also be used in other formulations such as scheduling in high-level synthesis for optimizing register usage and bus usage [7].

A. Related Works

A number of heuristic algorithms have been proposed for temporal partitioning. They include a list-scheduling-based algorithm in [21], a force-directed scheduling algorithm in [4], [5], a network-flow-based algorithm in [15], and a probability-based iterative-improvement algorithm in [6]. Recently, an exact integer linear programming formulation of the problem was given in [22]. We note that the integer linear programming approach can achieve better results at the expense of much larger runtime, and is feasible only for small circuit size. But none of these works consider temporal logic replication. Here, we propose to apply temporal logic replication after a prepartition is found, hence, it is compatible with all previously proposed temporal partitioning algorithms. Nevertheless, we also designed a new efficient hierarchical flow-based algorithm for computing prepartitions without replication in this paper. We show that our hierarchical flow-based algorithm compares favorably with the previously proposed algorithms. A preliminary version of this work was presented at ISPD'02 [16].

B. Paper Organization

The rest of the paper is organized as follows. In Section II, we give the formulation of the temporal partitioning problem for DRFPGA. In Section III, we present a hierarchical flow-based method to compute a performance-driven temporal partition by careful net modeling for temporal constraint satisfaction and buffer requirement counting. In Section IV, we define the min-area min-cut replication problem to optimally reduce the communication cost given a K-stage temporal partition satisfying all temporal constraints. We present an optimal algorithm to solve the min-area min-cut replication problem. We also present a flow-based replication heuristic in case that there is a tight area bound that limits the amount of replication. The experimental results are reported in Section V. We conclude the paper in Section VI.

II. Problem Formulation

Different architectures have been proposed for DRFPGA [8], [9], [17], [20]. The constraints imposed on partitioning are slightly different with different architectures. In this paper, we target our problem formulation on the Xilinx model [20]. However, we emphasize that one can easily modify the formulation and the algorithms to be presented in the subsequent sections for other architectures.

We follow the formulation and notation used in [6], [15] for temporal partitioning under the Xilinx model. A *user cycle* is a cycle that passes through all stages once (see Fig. 2). Given a circuit, we distinguish between two types of nodes in the circuit: *combinational nodes (C-nodes)* and *flip-flop nodes (FF-nodes)*. Note that a combinational circuit has combinational nodes only but a sequential circuit has both combinational nodes and flip-flop nodes. The following rules are given in [21] which must be followed when a circuit is partitioned for implementation on a DRFPGA to ensure the correctness of computation in the Xilinx model.

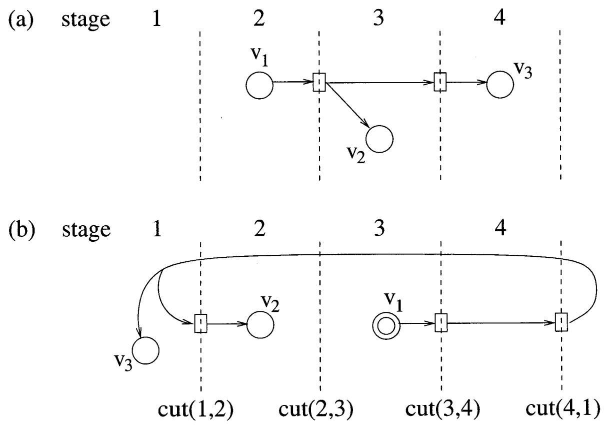


Fig. 4. (a) Storage required by a C-type net. (b) Storage required by an FF-type net.

1) Each combinational node must be scheduled in a stage no earlier than any of its fanin combinational nodes.
2) Each flip-flop node must be scheduled in a stage no earlier than any of its fanin combinational nodes.
3) Each flip-flop node must be scheduled in a stage no earlier than any of its fanout nodes. (This guarantees that all nodes that use the value of the flip-flop will use the value computed in the previous user cycle.)

The above rules can be summarized into two constraints as follows. Let $u \preceq v$ denote the temporal constraint that node u must be scheduled no later than node v. For all net $n = (v_1, \{v_2, \ldots, v_p\})$, where v_1 is the source terminal of the net, we have

- if v_1 is a C-node, then $v_1 \preceq v_j$ for $2 \leq j \leq p$ (1)

- if v_1 is an FF-node, then $v_j \preceq v_1,$ for $2 \leq j \leq p.$ (2)

If the source terminal v_1 of a net is a C-node, we call the net a *C-type net*. If the source terminal v_1 of a net is an FF-node, we call the net an *FF-type net*. For a C-type net, its datum will be used in the same user cycle that it is generated. It has to be buffered from the stage where its source terminal is assigned to till the last stage any of its other terminals is assigned to. See Fig. 4(a) for an example. For an FF-type net, its datum will be used in the next user cycle after its generation. Hence, it must be buffered in the current user cycle from the stage where its source terminal is assigned to all the way to the end of the current user cycle, and must remain buffered at the final stage of the current user cycle, and then in the next user cycle, it must be buffered from the first stage till the last stage that any of its other terminals is assigned to. [See Fig. 4(b) for an example.]

The total communication cost at the end of a stage is counted as follows. For a C-type net $(v_1, \{v_2, \ldots, v_p\})$, it incurs a communication cost of 1 at the end of each stage i such that $s(v_1) \leq i < \max_{2 \leq j \leq p} s(v_j)$, where $s(v)$ denotes the stage that node v is assigned to. For an FF-type net $(v_1, \{v_2, \ldots, v_p\})$, it incurs a communication cost of 1 at the end of each stage i such that $s(v_1) \leq i \leq K$ or $1 \leq i < \max_{2 \leq j \leq p} s(v_j)$, where K is the final stage. We note that the total communication cost at the end of stage K is always equal to the total number of FF-nodes in the circuit.

III. Hierarchical Flow-Based Temporal Partitioning

A K-stage temporal partition can be obtained by bipartitioning a circuit recursively. An approach using network flow computation was first

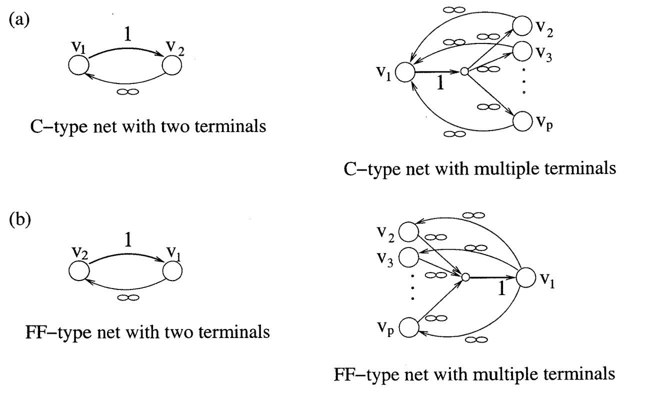

(a)
C–type net with two terminals
C–type net with multiple terminals
(b)
FF–type net with two terminals
FF–type net with multiple terminals

Fig. 5. Net modeling in [15].

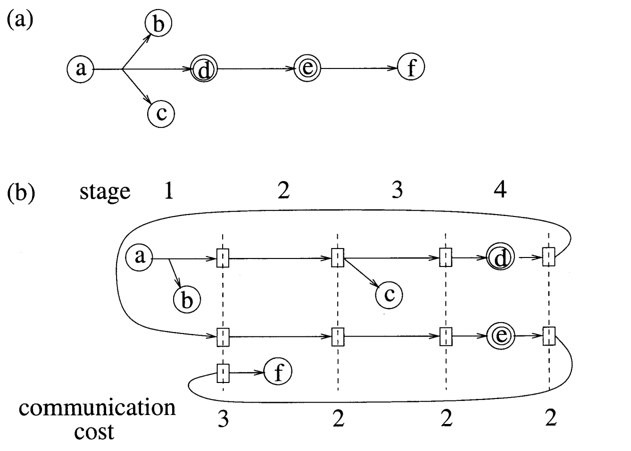


Fig. 6. Simple sequential circuit and a possible temporal partitioning of the circuit. (C-type net: $(a, \{b, c, d\})$. FF-type nets: $(d, \{e\})$, $(e, \{f\})$.).

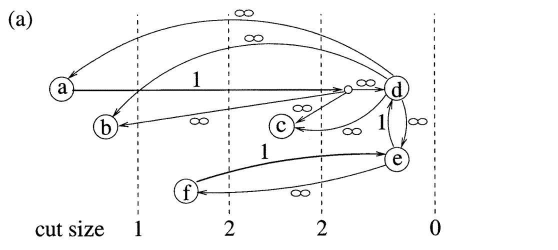
used by Liu and Wong [15]. However, there is a pitfall in the modeling of an FF-type net in [15] that though it correctly enforces the temporal constraints, it will lead to an incorrect estimation of the communication cost when the circuit is partitioned. We explain this problem in this section and give a correct modeling which ensures that the communication cost at each stage will be counted correctly during partitioning. In addition, instead of performing the bipartitionings in a *sequential* manner as in [15], we propose to perform the bipartitionings in a *hierarchical* manner. Finally, we consider how to compute a performance-driven temporal partition.

A. Net Modeling

A network-flow-based approach is a simple attractive approach to solve the temporal partitioning problem because it can easily handle temporal constraints by suitable network modeling. If there exists a temporal constraints $u \preceq v$, meaning that node u has to be scheduled to a stage no later than that of node v, we can model this constraint by introducing a directed arc (v, u) from v to u with infinite cost in the flow network. Recall that for a weighted directed graph, the cost of a (unidirectional) cut $(X, \bar{X})$ ($X \cap \bar{X} = \phi$ and $X \cup \bar{X} =$ vertex set of the graph) is the sum of the weights of all edges going from X to $\bar{X}$ [1]. So for any finite cut $(X, \bar{X})$ computed in the network, either we have (i) $u, v \in X$, (ii) $u, v \in \bar{X}$, or (iii) $u \in X$ and $v \in \bar{X}$, but we will never have $v \in X$ and $u \in \bar{X}$ (otherwise, the cut would have an infinite cost due to arc (v, u)).

The net modeling used in [15] is shown in Fig. 5. Though the modeling correctly enforces the temporal constraints [(1) and (2) in Section II] for both C- and FF-type nets, it does not account for the communication cost due to FF-type nets correctly. This is illustrated by the following example. Fig. 6(a) shows a sequential circuit with two flip-flop nodes. It is clear that the execution of the circuit takes three user clock cycles. A possible temporal partitioning of the circuit is shown in Fig. 6(b). Note that the output of flip-flop d at the first user cycle will be buffered and read by e at the second user cycle, and the output of flip-flop e at the second user cycle will be buffered and read by f at the third user cycle. It can be seen that the communication costs at the end of stages 1, 2, 3, and 4 are 3, 2, 2, and 2, respectively. However, if we use the net modeling shown in Fig. 5, the communication costs will be incorrectly computed as 1, 2, 2, and 0, respectively, as shown in Fig. 7(a).

Here, we present a new and correct modeling for FF-type net. Consider an FF-type net $n = (v_1, \{v_2, \ldots, v_p\})$. There are two possible conditions for which the net will incur a communication cost in cut$(i, i+1)(i = 1, 2, \ldots, K)$. First, if the source terminal v_1 is on the left hand side of cut$(i, i+1)$, net n will incur a cost of 1 in cut$(i, i+1)$ since its signal must be buffered at the end of stage i. (For example, the FF-net in Fig. 4(b) incurs a cost of 1 in both cut(3,4) and cut(4,1).) Second, if some terminal $v_j (2 \leq j \leq p)$ is on the right hand side of cut$(i, i+1)$, net n will incur a cost of 1 in cut$(i, i+1)$ since its signal must be buffered at the end of stage i. (For example, the FF-net in Fig. 4(b) incurs a cost of 1 in cut(1,2).) Fig. 8 shows our modeling for FF-type net. We make use of two artificial nodes s and t which we designate as the source node and the sink node of the flow network, respectively. Our modeling ensures that the size of cut$(i, i+1)$ is correctly increased by 1 when the source terminal v_1 is assigned to the left of cut$(i, i+1)$ [see Fig. 9(a)], or when some $v_j (2 \leq j \leq p)$ is assigned to the right of cut$(i, i+1)$ [see Fig. 9(b)], but is not affected by the net otherwise [see Fig. 9(c)]. Fig. 7(b) shows the size of each cut corresponding to the partition in Fig. 6(b) using this modeling. It can be seen that the cut sizes in Fig. 7(b) exactly match the communication costs in Fig. 6(b).



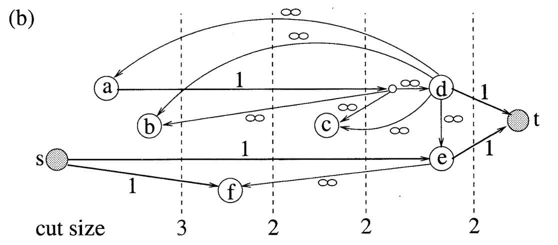


Fig. 7. (a) Cut sizes determined with the net model shown in Fig. 5. (b) Cut sizes determined with the corrected net model in Fig. 8.

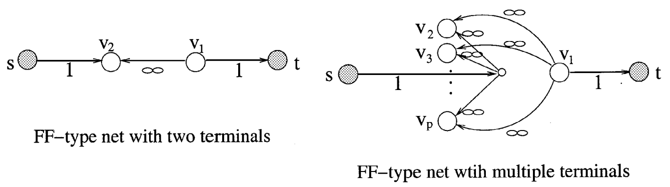

FF–type net with two terminals
FF–type net wtih multiple terminals

Fig. 8. Correct modeling of an FF-type net.

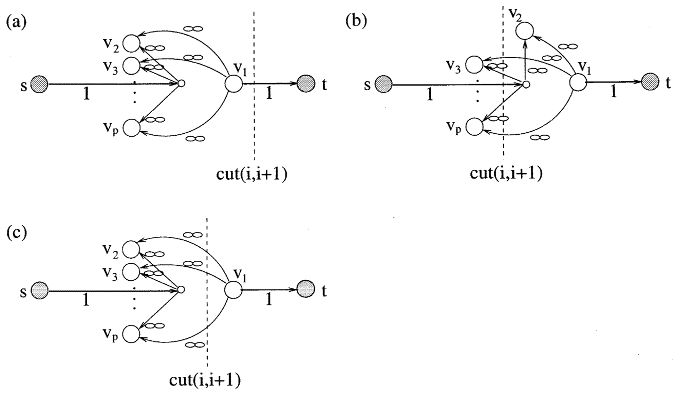


Fig. 9. Three cases for the cutting of an FF-type net $(\boldsymbol{v}_1, \{\boldsymbol{v}_2, \ldots, \boldsymbol{v}_p\})$. (a) If $\boldsymbol{v}_1$ is on the LHS of cut$(\boldsymbol{i}, \boldsymbol{i}+\mathbf{1})$, it increases the size of cut$(\boldsymbol{i}, \boldsymbol{i}+\mathbf{1})$ by 1. (b) If some $\boldsymbol{v}_j$ $(\boldsymbol{j} = \mathbf{2}, \ldots, \boldsymbol{p})$ is on the RHS of cut$(\boldsymbol{i}, \boldsymbol{i}+\mathbf{1})$, it increases the size of cut$(\boldsymbol{i}, \boldsymbol{i}+\mathbf{1})$ by 1. (c) If $\boldsymbol{v}_1$ is on the RHS of cut$(\boldsymbol{i}, \boldsymbol{i}+\mathbf{1})$ and $\boldsymbol{v}_j$ is on the LHS of cut$(\boldsymbol{i}, \boldsymbol{i}+\mathbf{1})$ for all $\boldsymbol{j} = \mathbf{2}, \ldots, \boldsymbol{p}$, the size of cut$(\boldsymbol{i}, \boldsymbol{i}+\mathbf{1})$ is not affected by the net.

B. Area-Balanced Partitions

With the correct net modeling, we can bipartition a circuit by bipartitioning its corresponding directed network using a heuristic algorithm, FBB, proposed by Yang and Wong [24]. It is an efficient max-flow min-cut heuristic that repeatedly cuts the oversized side with gradually increased cut sizes until the ratio of the areas of the two sides is within a desired range. The FBB algorithm is given in Fig. 10. In the algorithm, $w(X)$ denote the area of side X. It was shown in [24] that the repeated max-flow min-cut process can be implemented efficiently using incremental flow computation so that it has the same asymptotic time complexity as just one max-flow computation, i.e., $O(|V||E|)$.

Note that in the original FBB algorithm in [24], two circuit nodes are picked randomly as the source node s and the sink node t so as to bipartition the circuit into two parts X and $\bar{X}$ such that $s \in X$ and $t \in \bar{X}$, but here we do something different to compute a temporal partitioning. As explained in Section III-A, for temporal partitioning, we add two extra nodes as the source node s and sink node t, and connect them with the terminals of every FF-type net in a specific way. Moreover, as we will explain later in Section III-D, we will collapse all nodes preassigned to any stage between 1 to i into node s and collapse all nodes preassigned to any stage between $i+1$ to K into node t when we compute cut$(i, i+1)$ in order to obtain a performance-driven temporal partition.

C. Hierarchical Versus Sequential Bipartitioning

There are two possible ways to obtain a K-stage temporal partition by recursive bipartitioning. One possibility is to first bipartition the circuit into two parts of roughly equal sizes, then the two subcircuits are recursively bipartitioned in the same way until each subcircuit can fit into a stage. Another possibility is to apply the first bipartitioning to determine the first stage, then the rest of the circuit is recursively bipartitioned to obtain the second stage, the third stage, etc., in sequential order. We refer to the former as *hierarchical bipartitioning* and the latter as *sequential bipartitioning*.

We adopt the hierarchical bipartitioning approach even though the sequential bipartitioning approach was used in [15]. Our choice is motivated by an observation of the close relationship between the minimum communication temporal partitioning problem and the minimum cut linear arrangement problem. The minimum cut linear arrangement problem is described as follows. Given an undirected graph $G = (V, E)$ with n vertices, find a one-to-one mapping $s : V \rightarrow \{1, 2, \ldots, n\}$ so as to minimize

```
ALGORITHM FBB

Inputs: A directed graph model $G = (V, E)$ of a circuit using appropriate net modelling. The lower
        bound $lr$ and upper bound $ur$ on the ratio of the areas of the two sides of the partition.

Output: A feasible bipartition with a small cut size.

  1. Compute a maximum flow from $s$ to $t$. Let $X$ be the set of nodes reachable from $s$ through
     augmenting path, and $\bar{X} = V - X$. $C = (X, \bar{X})$.
  2. If $lr \le w(X)/w(\bar{X}) \le ub$ then stop and return $C$ as the answer.
  3. If $w(X)/w(\bar{X}) \le lr$ then
     3.1 collapse all nodes in $X$ to $s$;
     3.2 collapse to $s$ a node $v \in \bar{X}$ incident on a net in $C$;
     3.3 goto 1.
  4. If $w(X)/w(\bar{X}) \ge ub$ then
     4.1 collapse all nodes in $\bar{X}$ to $t$;
     4.2 collapse to $t$ a node $v \in X$ incident on a net in $C$;
     4.3 goto 1.
```

Fig. 10. The FBB algorithm for partitioning.

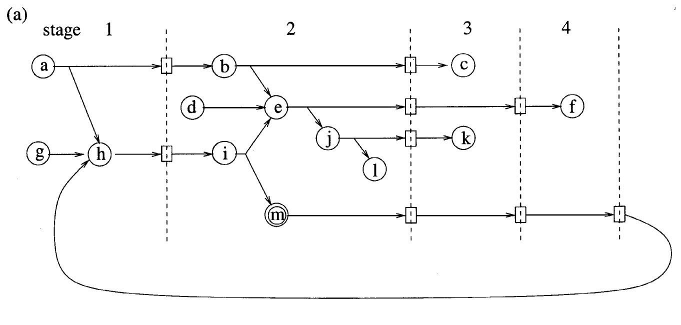
$\max_{i=1,2,\ldots,n-1}(\sum_{e \in S_i} c(e))$, where S_i denote the set of edges in E of the form (u, v) such that $s(u) \le i < s(v)$, and $c(e)$ denotes the cost of edge e. It is known that, if we have a ρ-approximation algorithm for graph bipartitioning, it can be applied hierarchically to derive a divide-and-conquer $O(\rho \log n)$-approximation algorithm for the minimum cut linear arrangement problem [19]. On the other hand, if the ρ-approximation graph bipartitioning algorithm is applied in a sequential manner, an $O(\rho n)$-approximation algorithm for the minimum cut linear arrangement problem will result. Now note that the minimum cut linear arrangement problem can be considered as a special case of the minimum communication temporal partitioning problem where the input hypergraph is a graph, the precedence constraint set is empty, and the number of stages is n.

D. Performance-Driven Temporal Partitioning

The period of a user cycle determines how fast an application can run on a DRFPGA. And the period of a user cycle depends on the temporal partitioning. The period of a user cycle is equal to $K \cdot E$, where K is the number of stages and E is the execution time of each stage. Since the number of stages K is fixed and is determined by the size of the circuit and the size of the device, we should target to compute a K-stage temporal partition that minimizes E. We note that E is determined by the stage with the maximum width, i.e., the stage with the maximum number of levels of combinational nodes. Thus, E is minimized when the widths of all stages are balanced.

So, when we bipartition a circuit for the first time, the lengths of the two halves of a combinational path on both sides should be upper bounded by $\lceil D/2 \rceil$[1] where D is the length of the longest combinational path in the circuit. Let $\delta_O(v)$ denote the maximum distance from combinational node v to a primary output or a flip-flop without passing through any other flip-flop. Let $\delta_I(v)$ denote the maximum distance from a primary input or a flip-flop to combinational node v without passing through any other flip-flop. When we first bipartition the circuit into $(X, \bar{X})$, any combinational node v with $\delta_I(v) > \lceil D/2 \rceil$ must be assigned to $\bar{X}$, otherwise there would be more than $\lceil D/2 \rceil$ levels of node delay in X. Similarly, any combinational node v with

[1]This upper bound can be relaxed minimally if there does not exist an area-balanced bipartition under the original bound.



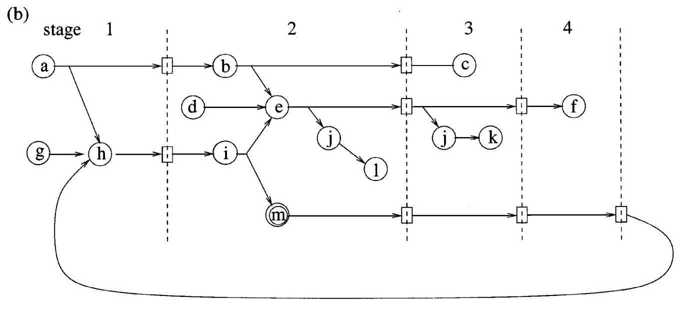


Fig. 11. Replication for communication cost reduction. (a) Before replicating node $\boldsymbol{j}$. (b) After replicating node $\boldsymbol{j}$. (C-type nets: $(\boldsymbol{a}, \{\boldsymbol{b}, \boldsymbol{h}\})$, $(\boldsymbol{b}, \{\boldsymbol{e}, \boldsymbol{c}\})$, $(\boldsymbol{d}, \{\boldsymbol{e}\})$, $(\boldsymbol{e}, \{\boldsymbol{f}, \boldsymbol{j}\})$, $(\boldsymbol{g}, \{\boldsymbol{h}\})$, $(\boldsymbol{h}, \{\boldsymbol{i}\})$, $(\boldsymbol{i}, \{\boldsymbol{e}, \boldsymbol{m}\})$, $(\boldsymbol{j}, \{\boldsymbol{l}, \boldsymbol{k}\})$. FF-type net: $(\boldsymbol{m}, \{\boldsymbol{h}\})$).

$\delta_O(v) > \lceil D/2 \rceil$ must be assigned to X, otherwise there would be more than $\lceil D/2 \rceil$ levels of node delay in $\bar{X}$.

In general, we will preassign a subset of nodes to their proper stages before partitioning for timing optimization. When we perform bipartitioning to compute cut$(i, i+1)$, all nodes that are preassigned to stages 1 to i are collapsed to the source node s of the network, and all nodes that are preassigned to stages $i+1$ to K are collapsed to the sink node t of the network. We note that this does not only guarantee the timing performance of the computed solution, it also reduces the running time of the partitioning process.

IV. Temporal Replication

Temporal logic replication exploits the slack logic capacity of a stage to reduce the communication cost. The degree of communication cost reduction by temporal replication depends on the amount of replication allowed, which in turn depends on the gate utilization per stage of the initial partition on the DRFPGA. We assume that a K-stage temporal partition without replication has been computed. The communication cost at the end of stage i is equal to the size of cut $(i, i+1)$. We can reduce the cut size by carefully replicating some nodes in stage i to stage $i+1$. For example, Fig. 11(a) shows a four-stage temporal partition without replication, the communication cost at the end of stage 2 can be reduced from 4 to 3 by replicating node j to stage 3 as shown in Fig. 11(b). Note that since we start with an original partition that already satisfies every temporal constraint, we do not have to worry about the temporal constraints when we perform replication. For example, in Fig. 11(b), the replica of node j in stage 3 does not need to precede node l because node l can get its correct input from the original copy of node j in stage 2.

Below, we define the min-cut replication problem and the min-area min-cut replication problem. Since there is an upper bound on the area of each stage in practice, it is desirable to minimize the amount of replication. We show that the min-area min-cut replication problem can be solved optimally by a flow-based algorithm. In case where the stage area bound is sufficiently large, it suffices to solve the min-area min-cut

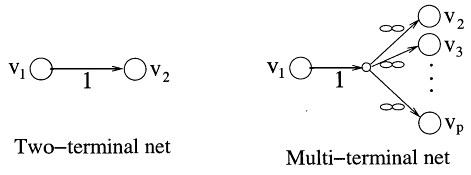


Fig. 12. Net modeling in network for replication set computation.

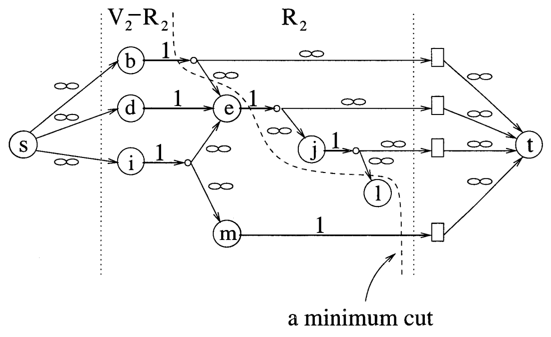


Fig. 13. Network for computing a replication set for stage 2 of the partition in Fig. 11(a).

replication problem. In case where it is not, we present a heuristic algorithm to compute a replication set to effectively reduce the communication cost without exceeding the stage area bound.

Min-Cut Replication Problem: Compute a subset of nodes in stage i for replication into stage $i+1$ such that, after replication, the communication cost at stage i is maximally reduced $(i = 1, 2, \ldots, K-1)$.[2]

Min-Area Min-Cut Replication Problem: Compute a minimum subset of nodes in stage i for replication into stage $i+1$ such that after replication the communication cost at stage i is maximally reduced $(i = 1, 2, \ldots, K-1)$.

We consider the min-area min-cut replication problem. Let V_i denote the set of nodes in stage i in the original partition before replication. Let R_i denote the set of nodes replicated from stage i to stage $i+1$. Observe that by replicating R_i into stage $i+1$, the original buffers required for buffering any output signals of R_i for stage $i+1$ can be removed (because R_i will also be in stage $i+1$ after replication), but new buffers are required to buffer any output signals of $V_i - R_i$ that are used by R_i in stage $i+1$. Hence, the min-area min-cut replication problem is equivalent to the problem of computing a minimum cut $(V_i - R_i, R_i)$ such that $|R_i|$ is minimized. We can solve this problem by using a flow-based method in a network $G_i' = (V_i', E_i')$. $V_i = V_i \cup B_i \cup \{s, t\}$ where B_i is the set of original buffers required at the end of stage i, and s and t are the source and sink nodes added for flow computation. Each net $(v_1, \{v_2, \ldots, v_p\})$ in stage i is modeled by a set of arcs in the form of a star[3] as shown in Fig. 12 so that the cut size is increased by 1 whenever the source terminal v_1 is in $V_i - R_i$ but some other terminal of the net is in R_i. There is an infinite capacity arc (b, t) for each node $b \in B_i$. Finally, there is an infinite capacity arc (s, v) for each node $v \in V_i$ that is a primary input [e.g., node d in Fig. 11(a)] or a node that receives any buffered input from the previous stage [e.g., nodes b and i in Fig. 11(a)]. (This is to avoid getting the trivial minimum cut solution $(V_i - R_i, R_i)$, where $R_i = V_i$.) Fig. 13 shows the network for computing a replication set for stage 2 of the partition in Fig. 11(a). A maximum flow from s to t can be computed for the constructed network G_i'. Taking $R_i = \{v \in V_i : \exists$ an augmenting path from v

[2]Note that the number of buffers required at the end of stage $\boldsymbol{K}$ is always equal to the number of flip-flop nodes in the circuit and cannot be reduced by replication.

[3]Note that temporal constraints can be safely ignored in the replication process.

```
ALGORITHM Min-cut Replication

Inputs: Stage index $i$ ($1 \leq i \leq K-1$). Stage area
    bound $U$.
Output: Replication set $R_i$ for replication from stage $i$
    to stage $i+1$.

    1. Construct replication network $G'_i$.
    2. Compute a maximum flow from $s$ to $t$. Let $R_i =$
       $\{v \in V_i : \exists$ an augmenting path from $v$ to $t\}$ and
       $X = V_i - R_i$.
    3. If $|V_{i+1}| + |R_i| \leq U$ then stop and return $R_i$.
    4. 4.1 Collapse all nodes in $X$ to $s$;
       4.2 Collapse to $s$ a node $v \in R_i$;
       4.3 Goto 2.
```

Fig. 14. The replication algorithm.

```
Communication Cost Reduction

    1. Identify the stage $i$ ($i = 1, \ldots, K-1$) s.t. the
       number of buffers required at the end of stage $i$
       is maximum.
    2. If replication has been performed from stage $i$ to
       stage $i+1$, then stop. Else perform replication
       from stage $i$ to stage $i+1$ and goto step 1.
```

Fig. 15. Procedure for reducing the communication cost of a $\boldsymbol{K}$-stage temporal partition by temporal replication.

to t in $G'_i\}$, we get a minimum cut $(V_i - R_i, R_i)$ such that $|R_i|$ is minimized [25]. In other words, we get a minimum replication set R_i such that the communication cost at stage i is maximally reduced.

If the stage area bound is sufficiently large, it suffices to solve the min-area min-cut replication problem as described above. If not, we can use the solution of the min-area min-cut replication problem as the starting point. Suppose R_i is the replication set computed for the min-area min-cut replication problem and $|V_{i+1}| + |R_i|$ exceeds the stage area bound. We can adapt the repeated max-flow min-cut process described in Section III-B to repeatedly cut the oversized replication set R_i to obtain smaller replication sets with gradually increased cut sizes until $|V_{i+1}| + |R_i|$ is within the required size. Our replication algorithm is shown in Fig. 14. It can be applied by the procedure in Fig. 15 to reduce the communication cost of any given K-stage temporal partition.

TABLE I
BENCHMARK CIRCUIT CHARACTERISTICS

Circuit	# Nodes	# Nets
c3540	1038	1016
c5315	1778	1655
c6288	2856	2824
c7552	2247	2140
s1423	831	750
s820	340	314
s838	495	459
s9234	6098	5846
s13207	9445	8653
s15850	11071	10385
s35932	19880	17830
s38417	25589	23845
s38584	22451	20719

TABLE II
RESULTS FOR EIGHT-STAGE PARTITIONING WITHOUT REPLICATION

Circuit	Max communication cost			Our Imp (%)	
	FBP-m	PAT	Ours	FBP-m	PAT
c3540	166	126	198	-19.28	-57.14
c5315	165	157	140	15.15	10.83
c6288	114	114	83	27.19	27.19
c7552	392	260	210	46.43	19.23
s820	81	43	52	35.80	-20.93
s838	71	72	70	1.41	2.78
s1423	120	106	101	15.83	4.72
s9234	502	430	381	24.10	11.40
s13207	901	838	683	24.20	18.50
s15850	877	808	761	13.23	5.82
s35932	2950	2138	2729	7.49	-27.64
s38417	2892	2628	2160	25.31	17.81
s38584	2796	3611	2275	18.63	37.00
average				18.11	3.81

V. EXPERIMENTAL RESULTS AND DISCUSSION

We implemented our flow-based replication algorithm for communication-cost reduction. We also implemented the hierarchical flow-based temporal partitioning algorithm for computing initial partitions without replication. We performed a number of experiments.

First, we performed a set of experiments to compare the performance of our hierarchical flow-based approach with two of the best heuristics reported in the literature [6], [15]. The first heuristic is FBP-m [15] which uses a sequential flow-based approach, and the second is probability-based algorithm for time-multiplied field programmable gate array (PAT) [6] which uses a probability-based iterative-improvement approach. As in [15] and [6], we applied our hierarchical flow-based temporal partitioning algorithm for balanced partitioning into eight stages such that the size of each stage is between $\lfloor 0.95n/8 \rfloor$ and $\lceil 1.05n/8 \rceil$ where n is the total number of nodes in the circuit. The same set of Microelectronics Center of North Carolina Partitioning'93 benchmark circuits were used as in [15] and [6]. The characteristics of the circuits are shown in Table I. The comparisons of the communication costs obtained by the three partitioning algorithms are shown in Table II. Our hierarchical flow-based partitioner outperformed FBP-m, a similar flow-based partitioner but carries out bipartitionings in a sequential manner, for all benchmark circuits but c3540. We have examined why the hierarchical flow-based partitioner did not produce a better result for c3540. When the hierarchical flow-based partitioner first divided c3540 into two subcircuits with the size of each subcircuit constrained to be between $4 \times \lfloor 0.95n/8 \rfloor$ and $4 \times \lceil 1.05n/8 \rceil$, it turned out that the best cut computed has $4 \times \lfloor 0.95n/8 \rfloor$ nodes on one side and $4 \times \lceil 1.05n/8 \rceil - 2$ nodes on the other. Hence, it limited the flexibility in the subsequent recursive bipartitioning process. Compared with PAT which is a completely different approach for the NP-hard

TABLE III
COMMUNICATION COST REDUCTION BY REPLICATION. (C = MAXIMUM COMMUNICATION COST, %IMP = PERCENT OF IMPROVEMENT, %REP = PERCENT OF NODES REPLICATED)

Circuit	Without replication	With replication					
		$\alpha = 1.1$			$\alpha = 1.2$		
	C (sec)	C (sec)	%Imp	%Rep	C (sec)	%Imp	%Rep
c3540	198 (3.65)	194 (0.18)	2.02	0.48	184 (0.15)	7.07	1.83
c5315	140 (3.17)	129 (0.14)	7.86	0.67	119 (0.10)	15.00	1.91
c6288	83 (12.55)	63 (0.57)	24.10	4.41	63 (0.42)	24.10	5.60
c7552	210 (9.85)	176 (0.46)	16.19	3.12	170 (0.31)	19.05	5.52
s820	52 (0.25)	48 (0.01)	7.69	1.76	45 (0.04)	13.46	5.59
s838	70 (0.22)	67 (0.01)	4.29	1.41	66 (0.02)	5.71	2.63
s1423	101 (0.92)	95 (0.09)	5.94	5.66	94 (0.08)	6.93	9.51
s9234	381 (48.60)	369 (2.71)	3.15	0.66	341 (2.00)	10.50	1.87
s13207	683 (203.96)	669 (1.51)	2.05	2.55	669 (0.79)	2.05	4.27
s15850	761 (282.74)	699 (5.32)	8.15	3.59	678 (3.86)	10.91	6.50
s35932	2729 (717.12)	2636 (70.56)	3.41	2.48	2599 (59.65)	4.76	4.99
s38417	2160 (2028.46)	1941 (53.43)	10.14	2.25	1853 (31.08)	14.21	5.81
s38584	2275 (584.11)	2131 (85.77)	6.33	1.61	2024 (58.94)	11.03	4.06
average			7.79	2.36		11.14	4.62

temporal partition problem,[4] we are able to obtain better results for ten out of 13 benchmark circuits.

As pointed out at the beginning of Section IV, the degree of communication cost reduction by temporal logic replication depends on the gate utilization per stage of the initial partition on the DRFPGA. For experimental purposes, we simply assume that the area of each stage after replication can be increased to $\lceil \alpha n/8 \rceil$ for $\alpha = 1.1$ and $\alpha = 1.2$. The results are shown in Table III. All of the initial partitions were computed by our hierarchical flow-based partitioner such that each stage contains between $\lfloor 0.95n/8 \rfloor$ and $\lceil 1.05n/8 \rceil$ of the nodes. The running times are shown inside parentheses along with the maximum communication costs. Note that, in general, the running time of our replication algorithm reduces when α becomes larger. When the stage area bound is loosened, the repeated max-flow min-cut process is expected to reach the stage area bound earlier. The fifth column and the eighth column of Table III show the percentage of nodes that are actually replicated for $\alpha = 1.1$ and $\alpha = 1.2$, respectively. For $\alpha = 1.1$, on average the communication cost was reduced by 7.79% with only 2.36% of nodes replicated. For $\alpha = 1.2$, on average the communication cost was reduced by 11.14% with only 4.62% of nodes replicated. This confirms that temporal logic replication is an effective means to reduce the communication cost.

We have a few remarks about the flip-flop nodes. First, we note that intuitively it is favorable to assign an FF-node v to or close to the final stage. For the FF-type net with source terminal v, it always has to be buffered from the stage where v is assigned to until the final stage irrespective of where its other terminals are assigned (see Section II). Our experiments confirmed that the majority of FF-nodes are assigned to the final stage in the computed initial partitions without replication. Second, we note that if an FF-node v in stage i is found to be in the replication set R_i for replication from stage i to stage $i+1$ by the temporal replication algorithm in Section IV, then we may safely delete the copy of node v from stage i after replicating it to stage $i+1$. The reason is that we know that the output of v is not used by any other node in stage i directly (recall that the output of an FF-node will only be used in the next user cycle). In our experiments, such a scenario occurred only very infrequently. A few FF-nodes were found to be in the replication sets computed for circuits s35932 and s38417.

[4]Temporal partitioning is NP-hard because the general partitioning problem is a special case with the precedence constraint set being empty, and it is known that the general partitioning problem is NP-hard.

VI. CONCLUSION

In this paper, we introduced the concept of temporal logic replication for DRFPGA partitioning. We considered using temporal logic replication to effectively exploit the slack logic capacity of a stage to reduce the communication cost. We formulated the min-area min-cut replication problem and presented an optimal algorithm to solve it. For the case that there is a tight area bound that limits the amount of replication, we presented a flow-based replication heuristic. In addition, we showed a correct network flow model for partitioning sequential circuits temporally and proposed a new hierarchical flow-based performance-driven partitioner for computing initial partitions without replication.

ACKNOWLEDGMENT

The authors would like to thank Prof. Y.-W. Chang for helpful discussions on [6] and to the anonymous reviewers for pointing out a mistake in the original version and for their valuable comments which helped to improve the paper.

REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[2] N. Bhat, K. Chaudhary, and E. S. Kuh, "Performance-Oriented Fully Routable Dynamic Architecture for a Field Programmable Logic Device," Univ. Calif., Berkeley, Memo no. UCB/RELM93/42, 1993.

[3] J. Brown, D. Chen, I. Eslick, E. Tau, and A. DeHon, *DELTA: Prototype for a First-Generation Dynamically Programmable Gate Array*. Cambridge, MA: MIT Press, 1995.

[4] D. Chang and M. Marek-Sadowska, "Buffer minimization and time-multiplexed I/O on dynamically reconfigurable FPGAs," in *Proc. ACM Int. Symp. FPGA*, 1997, pp. 142–148.

[5] ——, “Partitioning sequential circuits on dynamically reconfigurable FPGAs,” *IEEE Trans. Comput.*, vol. 48, pp. 565–578, June 1999.

[6] M. C. T. Chao, G. M. Wu, I. H. R. Jiang, and Y. W. Chang, “A clustering and probability-based approach for time-multiplexed FPGA partitioning,” in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1999, pp. 364–368.

[7] G. De Micheli, *Synthesis and Optimization of Digital Circuits*. New York: McGraw-Hill, 1994.

[8] A. DeHon, “DPGA-coupled microprocessors: commodity IC’s for the early 21st century,” in *Proc. IEEE Workshop FPGAs Custom Comput. Mach.*, 1994, pp. 31–39.

[9] T. Fujii *et al.*, “A dynamically reconfigurable logic engine with a multi-context/multi-mode unified-cell architecture,” in *Proc. IEEE Int. Solid-State Circuits Conf.*, 1999, pp. 364–365.

[10] K. Furuta, T. Fujii, M. Motomura, K. Wakabayashi, and M. Yamashina, “Spatial-temporal mapping of real applications on a dynamically reconfigurable logic engine (DRLE) LSI,” in *Proc. IEEE Custom Integrated Circuits Conf.*, 2000, pp. 151–154.

[11] J. Hwang and A. El Gamal, “Min-cut replication in partitioned networks,” *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 96–106, Jan. 1995.

[12] D. Jones and D. M. Lewis, “A time-multiplexed FPGA architecture for logic emulation,” in *Proc. IEEE Custom Integrated Circuits Conf.*, 1995, pp. 495–498.

[13] C. Kring and A. R. Newton, “A cell-replicating approach to mincut-based circuit partitioning,” in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1991, pp. 2–5.

[14] R. Kužnar, F. Brglez, and B. Zajc, “A unified cost model for min-cut partitioning with replication applied to optimization of large heterogeneous FPGA partitions,” in *Proc. ACM Eur. Design Automation Conf.*, 1994, pp. 271–276.

[15] H. Liu and D. F. Wong, “Network flow based circuit partitioning for time-multiplexed FPGAs,” in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1998, pp. 497–504.

[16] W. K. Mak and E. F. Y. Young, “Temporal logic replication for dynamically reconfigurable FPGA partitioning,” in *Proc. ACM Int. Symp. Physical Design*, 2002, pp. 190–195.

[17] M. Motomura, Y. Aimoto, A. Shibayama, Y. Yabe, and M. Yamashina, “An embedded DRAM-FPGA chip with instantaneous logic reconfiguration,” in *Proc. Symp. VLSI Circuits*, 1997, pp. 55–56.

[18] T. Nishitani, “An approach to a multimedia system on a chip,” in *Proc. IEEE Workshop Signal Processing Syst.*, 1999, pp. 13–22.

[19] D. B. Shmoys, “Cut problems and their application to divide-and-conquer,” in *Approximation Algorithms for NP-hard Problems*, D. S. Hochbaum, Ed. Boston, MA: PWS, 1997, pp. 192–235.

[20] S. Trimberger, “A time-multiplexed FPGA,” in *Proc. IEEE Symp. Field-Programmable Custom Comput. Mach.*, 1997, pp. 22–28.

[21] ——, “Scheduling designs into a time-multiplexed FPGA,” in *Proc. ACM Int. Symp. FPGA*, 1998, pp. 153–160.

[22] G. M. Wu, J. M. Lin, and Y. W. Chang, “Generic ILP-based approaches for time-multiplexed FPGA partitioning,” *IEEE Trans. Computer-Aided Design*, vol. 20, pp. 1266–11 274, Oct. 2001.

[23] M. Yamashina and M. Motomura, “Reconfigurable computing: its concept and a practical embodiment using newly developed dynamically reconfigurable logic (DRL) LSI,” in *Proc. Asia South Pacific Design Automation Conf.*, 2000, pp. 329–332.

[24] H. Yang and D. F. Wong, “Efficient network flow based min-cut balanced partitioning,” in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1994, pp. 50–55.

[25] ——, “New algorithms for min-cut replication in partitioned circuits,” in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1995, pp. 216–222.