# Temporal Logic Replication for Dynamically Reconfigurable FPGA Partitioning

Wai-Kei Mak
Dept. of Computer Science and Engineering
University of South Florida
Tampa, Florida 33620-5399
wkmak@csee.usf.edu

Evangeline F.Y. Young
Dept. of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
fyyoung@cse.cuhk.edu.hk

## ABSTRACT

In this paper, we propose the idea of temporal logic replication in dynamically reconfigurable field-programmable gate array partitioning to reduce communication cost. Temporal logic replication has never been explored before. We define the min-area min-cut replication problem given a $k$-stage temporal partition satisfying all temporal constraints and devise an optimal algorithm to solve this problem. We have also devised a flow-based replication heuristic in case there is a tight area bound that limits the amount of replication. In addition, we will present a correct network flow model for partitioning sequential circuits temporally.

## Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids—*Layout*; J.6 [**Computer Applications**]: Computer-Aided Engineering—*Computer-aided design*

## General Terms

Algorithms, Design, Performance

## 1. INTRODUCTION

A very large circuit can be partitioned into a number of subcircuits implemented by a set of interconnected field-programmable gate arrays (FPGAs). This type of partitioning is known as spatial partitioning. However, new dynamically reconfigurable FPGAs (DRFPGAs) offer a new possibility. In this paper, we address the partitioning problem for DRFPGA with temporal logic replication for communication cost reduction.

Dynamically reconfigurable FPGAs allow dynamic reuse of logic blocks and wire segments by employing more than one on-chip SRAM bit to control them. This enables the execution of a big computational task that otherwise cannot be fitted into a FPGA by temporally partitioning the task on a DRFPGA[5].


*ISPD'02*, April 7-10, 2002, San Diego, California, USA.


To implement a large circuit 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. 1 shows a circuit partitioned into stages 1 to 4, the execution sequence will be $1, 2, 3, 4, 1, 2, 3, 4, \ldots$. In order 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 computations of those nodes.

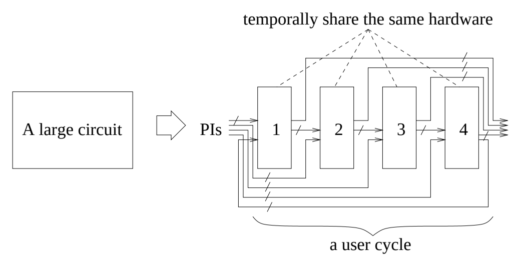


**Figure 1: Temporal Partitioning of a circuit.**

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. 2. The output of node $a$ has to be buffered at the end of stage 2 and should remain buffered until stage 4. It is known that the storage needed for buffering up signals creates a considerable overhead[2]. Hence an objective in temporal partitioning is to minimize the communication cost.

In spatial partitioning, it is known that logic replication can be performed to reduce the number of interconnections between components [7, 8, 6, 14]. However, replicating logic temporally has never been suggested or investigated before. In this paper, we consider using temporal logic replication to effectively exploit the slack logic capacity of a stage to reduce the communication cost. 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 this problem. We will also present a flow-based replication

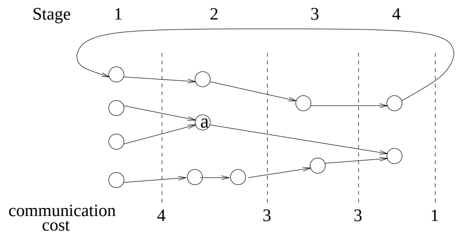


**Figure 2: Communication cost.**

heuristic in case there is a tight area bound that limits the amount of replication.

## 1.1 Related Works

A number of heuristic algorithms have been proposed for temporal partitioning. They include a list-scheduling based algorithm in [12], a force-directed scheduling algorithm in [2, 3], a network-flow based algorithm in [9], and a probability-based iterative-improvement algorithm in [4]. Recently, an exact integer linear programming formulation of the problem was given in [13]. 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 pre-partition 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 pre-partitions without replication in this paper. It is found that our hierarchical flow-based algorithm compares favorably with the previously proposed algorithms.

## 1.2 Paper Organization

The rest of the paper is organized as follows. In Section 2, we will formulate the temporal partitioning problem for DRFPGA. In Section 3, we will present a hierarchical flow-based method to compute a temporal pre-partition. In Section 4, we define the min-area min-cut replication problem given a $k$-stage temporal partition satisfying all temporal constraints and we will present an optimal algorithm to solve this problem. We will also present a flow-based replication heuristic in case there is a tight area bound that limits the amount of replication. Experimental results will be reported in Section 5 and we will conclude the paper in Section 6.

# 2. PROBLEM FORMULATION

Different architectures [5, 11] have been proposed for DRFPGA. In this paper, we target our problem formulation on the Xilinx model [11]. However, we emphasize that one can easily modify the formulation and our algorithms for other architectures.

We follow the formulation and notation used in [9, 4] for temporal partitioning under the Xilinx model. A *user cycle* is a cycle that passes through all stages (see Fig. 1). 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 must be followed when a circuit is partitioned for implementation on a DRFPGA to ensure the correctness of the computations:

1. Each combinational node must be scheduled in a stage no later than any of its fanout nodes.

2. Each flip-flop node must be scheduled in a stage no earlier than any of its fanin 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 using 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 $2 \preceq j \preceq p$ (1)
- if $v_1$ is a FF-node, then $v_j \preceq v_1$ for $2 \preceq j \preceq q$ (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 a FF-node, we call the net a *FF-type net*. For a C-type net, its datum will be used in same user cycle that it is generated. It has to be buffered from the stage where its source terminal is assigned to the last stage where any of its other terminals is assigned to. See Fig. 3(a) for an example. For a 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 from the first stage of the next user cycle till the last stage where any of its other terminals is assigned to. See Fig. 3(b) for an example.

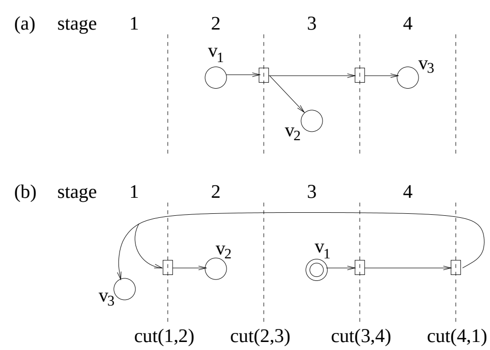


**Figure 3: (a) Storage required by a C-type net. (b) Storage required by a FF-type net.**

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 a 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 $i <$

$\max_{2 \le j \le p} s(v_j)$ where $k$ is the total number of stages. 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.

# 3. 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 proposed by Liu and Wong [9]. However, there is a pitfall in the modelling of a FF-type net in [9] that though it correctly enforces the temporal constraints, it will underestimate the communication cost when the circuit is bipartitioned recursively. We will explain this problem in subsection 3.1 and will give a correct modelling which ensures that the communication cost at each stage will be counted correctly when the circuit is recursively bipartitioned. In addition, we will also show that performing the bipartitionings in a *hierarchical* manner will give a better performance guarantee than performing the bipartitionings in a *sequential* manner as in [9].

## 3.1 Net Modelling

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 modelling. 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 $X \cup \bar{X}$ = the vertex set of the graph) is the sum of the weights of all the edges going from $X$ to $\bar{X}$[1]. Therefore for any finite cut $(X, \bar{X})$ computed in the network, either we have (i) $u, v \in X$, or (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 infinite cost due to arc $(v, u)$).

The net modelling used in [9] for computing a bipartition of a subcircuit is shown in Fig. 4. Though the modelling in Fig. 4 correctly enforces the temporal constraints ((1) and (2) in Section 2) for both C-type nets and FF-type nets, it does not count the communication cost due to FF-type nets correctly. Consider a FF-type net $n = (v_1, \{v_2, \ldots, v_p\})$. There are two possible conditions in which the net will incur a communication cost in $\text{cut}(i, i+1)$ $(i = 1, 2, \ldots, k)$. First, if the source terminal $v_1$ is on the left hand side of $\text{cut}(i, i+1)$, net $n$ will incur a cost of one in $\text{cut}(i, i+1)$ since its signal must be buffered at the end of stage $i$. For example, the FF-net in Fig. 3(b) incurs a cost of one in both cut(3,4) and cut(4,1). Second, if some terminal $v_j (2 \le j \le p)$ is on the right hand side of $\text{cut}(i, i+1)$, net $n$ will incur a cost of one in $\text{cut}(i, i+1)$ since its signal must be buffered at the end of stage $i$. For example, the FF-net in Fig. 3(b) incurs a cost of one in cut(1,2). However, it can be checked that the communication cost is not correctly accounted for by using the FF-type net modelling shown in Fig. 4(b).

Here we present a new and correct modelling for FF-type nets in Fig. 5. Our modelling ensures that the size of $\text{cut}(i, i+1)$ is correctly increased by 1 when the source terminal $v_1$ is assigned to the left of $\text{cut}(i, i+1)$ (see Fig. 6(a)), or when some $v_j$ $(2 \le j \le p)$ is assigned to the right of $\text{cut}(i, i+1)$ (see Fig. 6(b)), but is not affected by the net otherwise (see Fig. 6(c)).

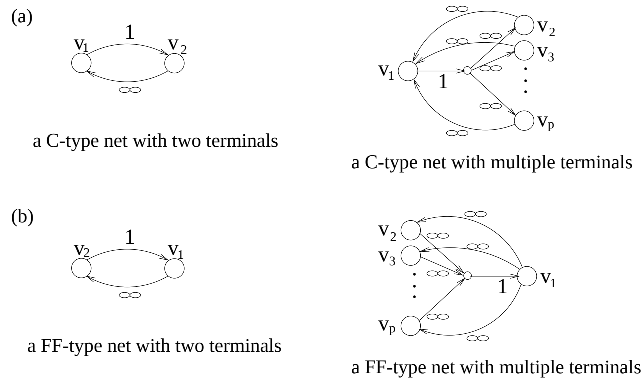


**Figure 4: Net modelling in [9].**

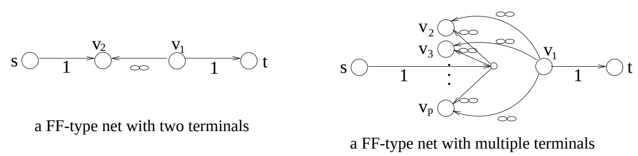


**Figure 5: Correct modelling of a FF-type net. Nodes $s$ and $t$ are the source and sink nodes of the constructed network.**

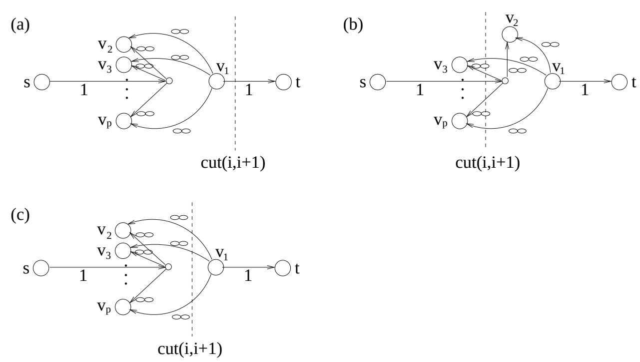


**Figure 6: Cutting of a FF-type net $(v_1, \{v_2, \ldots, v_p\})$. (a) If $v_1$ is on the left of $\text{cut}(i, i+1)$, it increases the size of $\text{cut}(i, i+1)$ by 1. (b) If some $v_j$ $(j = 2, \ldots, p)$ is on the right of $\text{cut}(i, i+1)$, it increases the size of $\text{cut}(i, i+1)$ by 1. (c) If $v_1$ is on the right of $\text{cut}(i, i+1)$ and $v_j$ is on the left of $\text{cut}(i, i+1)$ for all $j = 2, \ldots, p$, the size of $\text{cut}(i, i+1)$ is not affected by the net.**

## 3.2 Area-balanced Partitions

With the correct net modelling, we can bipartition a circuit by bipartitioning its corresponding flow network using the bipartitioning heuristic FBB proposed by Yang and Wong[14]. It is an efficient max-flow min-cut heuristic that repeatedly cuts the oversized side with gradually increasing cut sizes until the ratio of the areas of the two sides is within a desired range. It was shown in [14] 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., $O(|V||E|)$.

### 3.3 Hierarchical vs Sequential Bipartitioning

There are two possible ways to obtain a $k$-way 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 manner until each subcircuit can be fitted into a stage. Another possibility is to use the first bipartitioning to determine the first stage, then the rest of the circuit is repeatedly 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 adopted in [9]. The hierarchical bipartitioning approach can yield superior $k$-stage partition solutions in comparison with the sequential bipartitioning approach. In particular, it can be proved that if we apply a $\rho$-approximation bipartitioning algorithm in a hierarchical manner, the maximum communication cost of the resultant $k$-stage partition is upper bounded by $O(\rho \log k) \cdot r^*$ where $r^*$ is the maximum communication cost in an optimal $k$-stage partition. However if we apply the same bipartitioning algorithm in a sequential manner, the maximum communication cost of the resultant $k$-stage partition is upper bounded by $O(\rho k) \cdot r^*$. The same result is known for a similar problem, the minimum cut linear arrangement problem (see [10]), and can be proved similarly.

### 3.4 Timing Optimization

In order to minimize the execution time of a stage, we should balance the widths of all stages. Therefore when we first bipartition a circuit, the lengths of the longest paths on both sides should be upper bounded by $\lceil D/2 \rceil$ [1] where $D$ is the length of the longest path in the circuit.

Let $\delta_O(v)$ denote the length of the longest path from node $v$ to some primary output and $\delta_I(v)$ denote the length of the longest path from some primary input to node $v$. When we first bipartition the circuit into $(X, \bar{X})$, any node $v$ with $\delta_I(v) > \lceil D/2 \rceil$ must be assigned to $\bar{X}$, otherwise there would be a path of length greater than $\lceil D/2 \rceil$ in $X$. Similarly, any node $v$ with $\delta_O(v) > \lceil D/2 \rceil$ must be assigned to $X$, otherwise there would be a path of length greater than $\lceil D/2 \rceil$ in $\bar{X}$. In general, a subset of nodes can be pre-assigned to their proper stages before partitioning. So when we perform bipartitioning to compute $\mathrm{cut}(i, i+1)$, all nodes that are pre-assigned to stages 1 to $i$ are collapsed to the source node $s$ of the network, and all nodes that are pre-assigned 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.

## 4. TEMPORAL REPLICATION

Temporal logic replication exploits the slack logic capacity of a stage to reduce the communication cost. The degree of the 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 pre-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. 7(a) shows a 4-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. 7(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. 7(b), the replica of node $j$ in stage 3 does not need to precede node $l$ because node $l$ can gets its correct input from the original copy of node $j$ in stage 2.

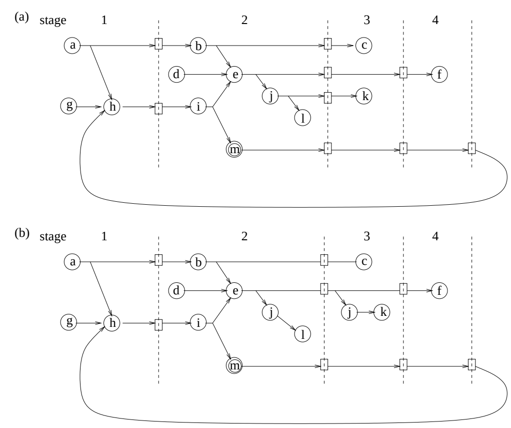


**Figure 7: Replication for communication cost reduction. (a) Before replicating node $j$. (b) After replicating node $j$. (C-type nets: $(a, \{b, h\})$, $(b, \{e, c\})$, $(d, \{e\})$, $(e, \{f, j\})$, $(g, \{h\})$, $(h, \{i\})$, $(i, \{e, m\})$, $(j, \{l, k\})$ FF-type net: $(m, \{h\})$)**

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 the stage area bound is sufficiently large, it suffices to apply this algorithm that solves the min-area min-cut replication problem optimally. In case it is not, we have also devised a heuristic algorithm to compute replication sets 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 ($i = 1, \ldots, k-1$ [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 ($i = 1, \ldots, k-1$).

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 $R_i$ be the set of nodes replicated from stage $i$ to stage $i+1$. Observe that by replicating $R_i$

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

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

into stage $i+1$, the original buffers required for buffering up the 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 signal of $V_i - R_i$ that is 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 modelled by a set of arcs in the form of a star[3] as shown in Fig. 8 so that the cut size is increased by 1 whenever the source terminal $v_1$ is in $V_i - R_i$ but some other terminals of the net are 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. 7(a)) or a node that receives any buffered input from the previous stage (e.g. nodes $b$ and $i$ in Fig. 7(a)). This is to avoid getting the trivial minimum cut solution $(V_i - R_i, R_i)$ where $R_i = V_i$. Fig. 9 shows the flow network for computing a replication set for stage 2 of the partition in Fig. 7(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 $v$ to $t$ in $G'_i\}$, we get a minimum cut $(V_i - R_i, R_i)$ such that $|R_i|$ is minimized[15]. In other words, we get a minimum replication set $R_i$ such that the communication cost at stage $i$ is maximally reduced.

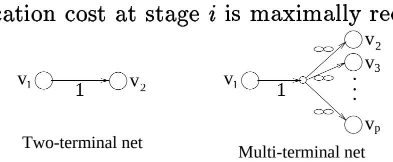


**Figure 8: Net modelling for replication set computation.**

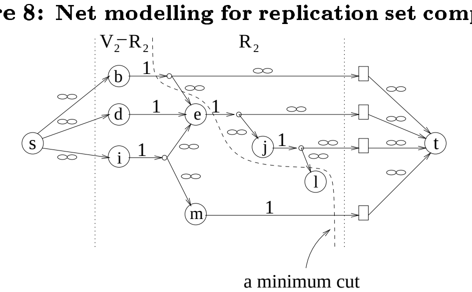


**Figure 9: Network for computing a replication set for stage 2 of the partition in Fig. 7(a).**

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 but $|V_{i+1}| + |R_i|$ exceeds the stage area bound. We can adapt the repeated max-flow min-cut process described in Section 3.2 to repeatedly cut the oversized replication set $R_i$ to obtain smaller replication sets with gradually increasing cut sizes until $|V_{i+1}| + |R_i|$ is within the required size. The replication algorithm is given below.

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

Min-cut Replication under Stage Area Bound

Inputs: Stage index $i$ $(1 \le i \le k-1)$. Stage area bound $A$.

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 $|V_{i+1}| + |R_i| \le A$ 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.

Given a pre-partition, we apply the following procedure to reduce the maximum communication cost.

Temporal Replication for 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 $i+1$, stop; otherwise, perform replication from stage $i$ to stage $i+1$ and goto step 1.

## 5. EXPERIMENTAL RESULTS

We implemented our flow-based replication algorithm for communication cost reduction and the hierarchical flow-based temporal partitioning algorithm for computing pre-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 literature [9, 4]. The first heuristic is FBP-m[9] which uses a sequential flow-based approach, and the second is PAT[4] which uses a probability-based iterative-improvement approach. As in [9] and [4], 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 MCNC Partitioning93 benchmark circuits were used as in [9] and [4]. The characteristics of the circuits are shown in Table 1. The results are shown in Table 2. Our hierarchical flow-based partitioner outperformed FBP-m, a similar flow-based partitioner but performing bipartitionings in a sequential manner, for all but one benchmark circuit. It also obtained better results than PAT for ten out of the thirteen benchmark circuits.

**Table 1: Benchmark circuit characteristics.**

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

As pointed out at the beginning of Section 4, the degree of communication cost reduction by temporal logic replication depends on the gate utilization per stage of the pre-partition on the DRFPGA. For experimental purpose, 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 re-

**Table 2: Results of 8-stage partitioning without replication.**

| Circuit | Max communication cost | | | Our Imprv (%) | |
|---|---|---|---|---|---|
| | 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 | 688 | 23.64 | 17.90 |
| s15850 | 877 | 808 | 761 | 13.23 | 5.82 |
| s35932 | 2950 | 2138 | 2729 | 7.49 | -27.64 |
| s38417 | 2892 | 2628 | 2194 | 24.14 | 16.51 |
| s38584 | 2796 | 3611 | 2280 | 18.45 | 36.86 |
| average | | | | 17.97 | 3.66 |

sults are shown in Table 3. All the pre-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 fifth column and the eighth column show the percentage of nodes that are actually replicated for $\alpha = 1.1$ and $\alpha = 1.2$, respectively. For $\alpha = 1.1$, the communication cost was reduced by 7.38% on average with only 2.18% of nodes replicated. For $\alpha = 1.2$, the communication cost was reduced by 10.94% on average with only 4.46% of nodes replicated.

**Table 3: Communication cost reduction by replication. (C = maximum communication cost, Imp = improvement, Rep = nodes replicated)**

| Circuit | Rep. No. | With replication | | | | | |
|---|---|---|---|---|---|---|---|
| | | $\alpha = 1.1$ | | | $\alpha = 1.2$ | | |
| | C | C | Imp (%) | Rep (%) | C | Imp (%) | Rep (%) |
| c3540 | 198 | 194 | 2.02 | 0.48 | 184 | 7.07 | 1.83 |
| c5315 | 140 | 129 | 7.86 | 0.67 | 119 | 15.00 | 1.91 |
| c6288 | 83 | 63 | 24.10 | 4.41 | 63 | 24.10 | 5.60 |
| c7552 | 210 | 176 | 16.19 | 3.12 | 170 | 19.05 | 5.52 |
| s820 | 52 | 48 | 7.69 | 1.76 | 45 | 13.46 | 5.59 |
| s838 | 70 | 67 | 4.29 | 1.41 | 66 | 5.71 | 2.63 |
| s1423 | 101 | 95 | 5.94 | 5.66 | 94 | 6.93 | 9.51 |
| s9234 | 381 | 369 | 3.15 | 0.66 | 341 | 10.50 | 1.87 |
| s13207 | 688 | 669 | 2.76 | 2.54 | 669 | 2.76 | 4.28 |
| s15850 | 761 | 699 | 8.15 | 3.59 | 678 | 10.91 | 6.50 |
| s35932 | 2729 | 2636 | 3.41 | 2.48 | 2599 | 4.76 | 4.99 |
| s38417 | 2194 | 2104 | 4.10 | 0.63 | 1957 | 10.80 | 3.64 |
| s38584 | 2280 | 2137 | 6.27 | 0.98 | 2025 | 11.18 | 4.12 |
| average | | | 7.38 | 2.18 | | 10.94 | 4.46 |

## 6. CONCLUSIONS

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 also presented a correct network flow model for partitioning sequential circuits temporally and devised a new hierarchical flow-based partitioner for computing pre-partitions satisfying all temporal constraints.

## Acknowledgement

We would like to thank Prof. Yao-Wen Chang for helpful discussion on [4].

## 7. REFERENCES

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

[2] D. Chang and M. Marek-Sadowska, "Buffer Minimization and Time-multiplexed I/O on Dynamically Reconfigurable FPGAs", in *Proc. of the ACM International Symposium on Field Programmable Gate Arrays*, pp. 142-148, 1997.

[3] D. Chang and M. Marek-Sadowska, "Partitioning Sequential Circuits on Dynamically Reconfigurable FPGAs", in *Proc. of the ACM International Symposium on Field Programmable Gate Arrays*, pp. 161-167, 1998.

[4] 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. of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 364-368, 1999.

[5] A. DeHon, "DPGA-coupled Microprocessors: Commodity ICs for the Early 21st Century", in *Proc. of IEEE Workshop on FPGAs for Custom Computing Machines*, pp. 31-39, 1994.

[6] J. Hwang and A. El Gamal, "Min-Cut Replication in Partitioned Networks", *IEEE Trans. on CAD*, vol. 14(1), pp. 96-106, Jan. 1995.

[7] C. Kring and A.R. Newton, "A Cell-replicating Approach to Mincut-Based Circuit Partitioning", in *Proc. of the IEEE International Conference on Computer-Aided Design*, pp. 2-5, 1991.

[8] 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. of the ACM European Design Automation Conf.*, pp.271-276, 1994.

[9] H. Liu and D.F. Wong, "Network Flow Based Circuit Partitioning for Time-Multiplexed FPGAs", in *Proc. of the IEEE International Conference on Computer-Aided Design*, pp. 497-504, 1998.

[10] D.B. Shmoys, "Cut Problems and Their Application to Divide-and-Conquer", *Approximation Algorithms for NP-hard Problems*, (D.S. Hochbaum, ed.) PWS, pp. 192-235, 1997.

[11] S. Trimberger, "A Time-Multiplexed FPGA", in *Proc. of IEEE Symposium on Field-Programmable Custom Computing Machines*, pp. 22-28, 1997.

[12] S. Trimberger, "Scheduling Designs into a Time-Multiplexed FPGA", in *Proc. of the ACM International Symposium on Field Programmable Gate Arrays*, pp. 153-160, 1998.

[13] G.M. Wu, J.M. Lin, and Y.W. Chang, "Generic ILP-Based Approaches for Time-Multiplexed FPGA Partitioning", *IEEE Trans. on CAD*, vol. 20(10), pp. 1266-11274, Oct. 2001.

[14] H. Yang and D.F. Wong, "Efficient Network Flow based Min-Cut Balanced Partitioning", in *Proc. of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 50-55, 1994.

[15] H. Yang and D.F. Wong, "New Algorithms for Min-Cut Replication in Partitioned Circuits", in *Proc. of the IEEE Int'l Conf. on Computer-Aided Design*, pp. 216-222, 1995.