

Power-Aware Technology Mapping for LUT-Based FPGAs

Jason H. Anderson and Farid N. Najm

Department of Electrical and Computer Engineering

University of Toronto

Toronto, Ontario, Canada M5S 3G4

janders@eecg.toronto.edu, f.najm@utoronto.ca

Abstract

We present a new power-aware technology mapping technique for LUT-based FPGAs which aims to keep nets with high switching activity out of the FPGA routing network and takes an activity-conscious approach to logic replication. Logic replication is known to be crucial for optimizing depth in technology mapping; an important contribution of our work is to recognize the effect of logic replication on circuit structure and to show its consequences on power. In an experimental study, we examine the power characteristics of mapping solutions generated by several publicly available technology mappers. Results show that for a specific depth of mapping solution, the power consumption can vary considerably, depending on the technology mapping approach used. Furthermore, results show that our proposed mapping algorithm leads to circuits with substantially less power dissipation than previous approaches.

1. Introduction

Field-programmable gate arrays (FPGAs) are a popular choice for digital circuit implementation because of their fast turnaround time, growing density and speed, and relatively low cost. State-of-the-art FPGAs have the capacity to implement millions of gates [1] and their application has migrated from being primarily a prototyping platform to their use in low to medium volume production designs. Despite continuing technology scaling and decreasing supply voltage values, the power consumed by the largest FPGA devices is increasing, with the power of the largest chips now being measured in watts [2]. Reducing the power consumption of FPGAs is beneficial as it leads to lower packaging and cooling costs as well as improves reliability. Additionally, if FPGAs are to be used more pervasively in portable battery-powered applications, low power consumption is essential.

The logic blocks in modern FPGAs are comprised of one or more look-up-tables (LUTs), registers, as well as arithmetic and other circuitry. An important step in the FPGA CAD flow is technology mapping, which involves

transforming a circuit from a generic form into a network of LUTs. LUT-based technology mapping has been studied extensively in recent years, with most research being focused on optimizing the area [3] and/or the depth [4,5] of the mapped circuit. Power represents the third, largely unexplored axis along which an FPGA design should be optimized.

Our focus in this paper is on optimizing depth and power in LUT-based technology mapping. We present a new technology mapping algorithm that allows one to explore the depth/power curve and to trade-off one criterion for the other. The paper is organized as follows: In the remainder of this section, we review the sources of power dissipation in FPGAs and briefly discuss previously published power-aware technology mapping algorithms. We present necessary background material in Section 2. Our technology mapping approach is described in Section 3. Section 4 presents our experimental study and results. Conclusions and suggestions for future work are offered in Section 5.

1.1. FPGA Power Dissipation

Several studies of FPGA power consumption have appeared recently in the literature [6,2]. These works have shown that power dissipation in FPGA devices is predominantly in the programmable interconnection network. In the Xilinx Virtex-II [1] family for example, it was reported that between 50–70% of total power is dissipated in the interconnect, with the remainder being dissipated in the clocking, logic and I/O blocks [6]. This breakdown differs substantially from custom ASIC technology in which clock distribution often dominates power consumption [7]. The difference in the sources of power dissipation between these two technologies lies in the composition of their interconnect structures: FPGA interconnect consists of pre-fabricated wire segments of various lengths, with used and unused routing switches attached to each wire segment.

The primary components of power dissipation in digital CMOS circuits are transistor leakage current (static power), short-circuit current and the charging and discharging of capacitance [7]. Leakage and short-circuit

current presently comprise only a small fraction of total FPGA power dissipation [6]. The majority of power dissipation results from charging and discharging capacitance and is characterized by:

$$P_{avg} = \frac{1}{2} \sum_{i \in nets} C_i \cdot f_i \cdot V^2 \quad (1)$$

where P_{avg} represents average power consumption, C_i represents the load capacitance of a net i , f_i represents the average toggle rate of net i (the *net switching activity*) and V is the voltage supply.

Our focus in this paper is on reducing the power dissipated in the FPGA interconnection network as computed by (1). In the remainder of the paper, we use the term *power* to refer to this interconnect power.

1.2. Power-Aware Technology Mapping

A number of power-aware technology mapping algorithms for LUT-based FPGAs have been proposed in recent years. Farrahi and Sarrafzadeh [8] presented a heuristic algorithm that minimized power at the expense of both area and depth. Their algorithm provides a 14% improvement over an algorithm that solely optimizes area. Li, Mak and Katkooi [9] described a power-aware mapping algorithm that produces depth-optimal mapping solutions and optimizes power in portions of a circuit that are not depth critical. Wang et al. [10] described an algorithm that focused on optimizing both power and area. They compared their approach to Farrahi and Sarrafzadeh's and found their approach able to reduce power by an additional 14%, while at the same time using fewer LUTs. These prior works each offer a "point solution" to power optimization in technology mapping; to our knowledge, no work has examined how power can be traded-off with other optimization criteria.

2. Preliminaries

Before presenting our technology mapping algorithm, we review some terminology. In this paper, we use terminology similar to that found in [4].

The combinational part of a logic circuit can be represented as a *Boolean network*, which is a directed acyclic graph (DAG) in which each node represents a single-output logic function and edges between nodes represent input/output dependencies between the corresponding logic functions. A primary input node is a node with an in-degree of 0; a primary output node has an out-degree of 0. For a node z in a circuit DAG, let $input(z)$ represent the set of nodes that are fanins of z and $output(z)$ represent the nodes that are fanouts of z . For a

subgraph, H , of a DAG, let $input(H)$ represent the set of nodes outside of H that are fanins of nodes in H ; let $output(H)$ be the set of nodes that are outside of H that are fanouts of nodes in H .

A node x is said to be a predecessor of node z if there exists a directed path in the graph from x to z . The subgraph consisting of a node z and *all* its predecessors will be referred to as *the subgraph rooted at z* . For any node z in a network, a *K -feasible cone* at z , F_z , is defined to be a subgraph consisting of z and *some* of its predecessors such that $|input(F_z)| \leq K$. A K -input LUT or K -LUT can implement any logic function with less than or equal to K inputs. Consequently, the technology mapping problem for K -LUTs can be thought of as "covering" an input Boolean network with K -feasible cones. Generally, there are many K -feasible cones for each node in the network, each having different area, delay or power characteristics.

A concept closely related to K -feasible cone is that of *K -feasible cut*. A K -feasible cut for a node z is a partition, (X, \bar{X}) , of the nodes in the subgraph rooted at z such that $z \in \bar{X}$ and the number of nodes in X that fanout to a node in \bar{X} is $\leq K$. Figure 1(a) shows a network having an output node z . The figure shows two 4-feasible cuts for node z . There is a one-to-one correspondence between K -feasible cuts and K -feasible cones: given a cut, (X, \bar{X}) , the K -feasible cone is simply the subgraph induced by the nodes in \bar{X} . The problem of finding all possible K -LUTs that generate a node z 's logic function is equivalent to the problem of enumerating all K -feasible cuts for node z .

To simplify the presentation of our algorithm, for a K -feasible cut, $C_z = (X, \bar{X})$, for a node z , we use $Nodes(C_z)$ to represent the set \bar{X} ($z \in \bar{X}$). We use $Support(C_z)$ to represent subset of nodes in X that fanout to a node in \bar{X} . For example, for cut 2 in Figure 1(a), $Nodes(cut\ 2) = \{c, z\}$ and $Support(cut\ 2) = \{b, i5, i6\}$. We use $Cuts(z)$ to represent the set of all feasible cuts for a node z .

2.1. Logic Replication and Power

Logic replication or duplication is performed implicitly when a LUT is used to implement a K -feasible cone that contains a node having a fanout outside that cone. It is widely known that logic replication is necessary for depth minimization. Consider again the network shown in Figure 1. Figure 1(b) shows a mapping solution without logic replication, assuming LUTs with 4 inputs, where LUTs are shown as shaded, dashed regions. The duplication-free mapping has depth 2 and uses 3 LUTs. Figure 1(c) shows a mapping solution in which logic replication is permitted. This solution has a depth of 1 which is achieved by replicating node b ; node b is covered by two different LUTs in the mapping solution.

When a node in a circuit is replicated for depth minimization, a connection from the node to one of its fanouts is “covered” within a LUT. In Figure 1(c) for example, both of the connections from node b to its fanouts are covered within LUTs. Such covered connections are not routed through the FPGA interconnection network and therefore do not contribute to interconnect power dissipation. The other consequence of node replication is that it generally increases the *fanout* of nodes that *fanin* to the replicated node. Referring again to Figure 1(b), primary inputs $i3$ and $i4$ each have one fanout LUT. In Figure 1(c), node b is replicated and therefore, primary inputs $i3$ and $i4$ must drive two LUTs. Thus, we see that replicating a node for depth minimization has two effects: 1) connections from the replicated node to its fanouts may be covered within LUTs and 2) the fanout of nodes that fanin to the replicated node is generally increased.

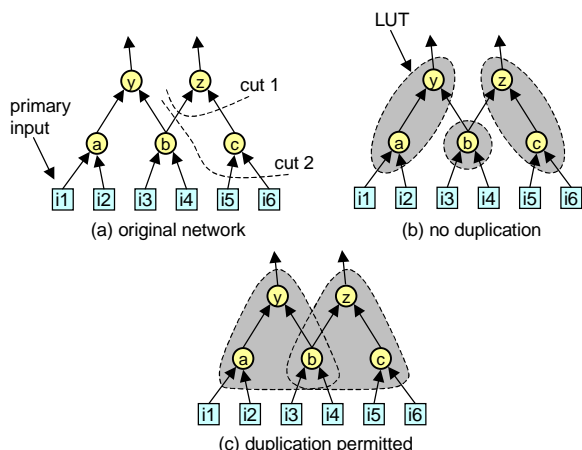


Figure 1. Illustration of feasible cuts; effect of logic replication in LUT mapping

Equation (1) specifies that power consumption depends linearly on switching activity. An important characteristic of switching activity in combinational circuits is that it typically decreases with circuit depth. Previous empirical research has shown in fact that activity falls quadratically with depth, on average [11]. This suggests that a node is likely to have fanins with higher switching activity than the switching activity at its output. Replicating a node for depth minimization covers a fanout of the node within a LUT, but increases the fanout of the node’s fanins. The activity/depth relationship implies that the activity on the signals whose fanout has been increased is likely higher than the activity on the signal whose fanout has been decreased (by covering a connection within a LUT). Consequently, we believe that logic replication in LUT mapping is generally undesirable from a power perspective, except in specific cases, depending on the switching activities local to a node. We apply this notion

in our technology mapping algorithm, where we introduce an activity-based penalty for the replication of a node.

3. Algorithm Description

Our technology mapping algorithm operates in three phases. The high-level flow of our approach is similar to that used in [12] and [10]. In phase 1, we construct the set of K -feasible cuts for each node in the network. In phase 2, we compute costs for the cuts generated in phase 1, and select a “best cut” for each node. In phase 3, we use the best-cost cuts and transform the Boolean network to produce the final LUT mapping solution. We now describe each phase in detail.

3.1. Generating K -Feasible Cuts

Traversing the network from primary inputs to primary outputs, the cuts for each node, z , are generated by merging cuts from its fanin nodes using the method described in [12,13]. At a high level, this works as follows: Consider a node z with two fanin nodes, a and b . The list of K -feasible cuts for a and b have already been computed, as a result of the network traversal order. Say node a has two K -feasible cuts, C_{a1} and C_{a2} , and node b has one K -feasible cut, C_b . We can merge C_{a1} and C_b to create a cut, C_{z1} , for node z such that $Support(C_{z1}) = Support(C_{a1}) \cup Support(C_b)$ and $Nodes(C_{z1}) = z \cup Nodes(C_{a1}) \cup Nodes(C_b)$. Clearly, if $|Support(C_{z1})| > K$, the resulting cut is not K -feasible and it is therefore discarded. Similarly, we can attempt to merge C_{a2} and C_b to create another cut, C_{z2} , for node z . This provides a general picture of how the cut generation procedure works; however, there are several special cases to consider, and the interested reader is referred to [13] for full details.

Note that other LUT-based technology mapping algorithms prune the cut-set for each node in order to reduce run-time [12]. Although an upper bound on the number of cuts for a node is $O(n^K)$, where n is the number of nodes in the circuit, we have observed that in actual circuits, the set of all cuts can be computed quickly when the target LUTs are small, as in commercial FPGAs. Thus, in our work, we did not find a need to implement pruning techniques such as [12], although this is certainly possible.

3.2. Costing Cuts

After computing the set of K -feasible cuts for each node in the network, we again traverse the network from primary inputs to primary outputs and select a “best cut” for each node. The “best cut” for each node is determined using a cost function with several components, reflecting

depth ($DCost$), power ($PCost$) as well as a *logic replication* cost ($RCost$), to be described below. For a node z with a K -feasible cut, C_z , we define the cost of the cut to be:

$$Cost(C_z) = \alpha \cdot DCost(C_z) + \beta \cdot PCost(C_z) + \gamma \cdot RCost(C_z) \quad (2)$$

where the parameters α , β and γ are coefficients reflecting the relative importance of each term. After computing the cost of the K -feasible cuts for node z , the best cut is selected to be the one with the minimum cost. We refer to the best cut for z as $BestCut(z)$.

We now elaborate on the terms of (2). The depth cost of a cut C_z is defined to be the depth of the LUT mapping solution of the subgraph rooted at node z , if $Nodes(C_z)$ is implemented as a LUT in the mapping solution. That is:

$$DCost(C_z) = 1 + \max_{v \in Support(C_z)} \{DCost(BestCut(v))\} \quad (3)$$

Thus, to compute the depth cost of cut C_z , we look at the depth cost of the best cut for each node, v , that fans out to a node in $Nodes(C_z)$. For each of these support nodes, the best cut has already been selected since we are traversing the network in an input-to-output fashion. Primary input nodes are assigned a depth cost of zero.

For power optimization, our goal is to keep high-activity connections out of the FPGA interconnect. Given this, we aim to “capture” as many high-activity connections as possible within LUTs, leaving only low-activity connections between LUTs. We therefore define the power cost of cut C_z for node z to be:

$$PCost(C_z) = \sum_{v \in Support(C_z)} [f_v + PCost(BestCut(v))] - \sum_{w \in Nodes(C_z)} [f_w \cdot |output(w) \cap Nodes(C_z)|] \quad (4)$$

where f_x represents the switching activity of the net driven by a node x . The first summation tallies the activities of the connections in the mapping solution of the subgraph rooted at z . The first term in the first summation represents the switching activities of nodes that fanout to a node in $Nodes(C_z)$. The nets driven by these nodes will need to be routed through the interconnect if $Nodes(C_z)$ is implemented as a LUT in the mapping solution; hence, they contribute to higher cost. The second term in the first summation represents the power cost of the mapping solutions rooted at each of the support nodes. The second summation term, whose sign is negative, represents the sum of the fanout-weighted switching activity on the

connections that have been captured inside a LUT if $Nodes(C_z)$ is implemented as a LUT. For each node w in $Nodes(C_z)$, it counts the number of w 's fanouts that are in $Nodes(C_z)$ and multiplies this count by the activity of the signal driven by w .

Prior to defining the replication cost term of (2), we introduce two additional concepts: First, we specify the *slack* and the *slack weight* for a node. Second, we present the notion of *replicated nodes*. The slack of a node z , $Slack(z)$, is defined to be the number of levels in the circuit's Boolean network DAG by which the depth of node z may be increased, without increasing the overall depth of the DAG. For example, if a node has slack 0, then its depth cannot be increased without also increasing the overall depth of the DAG. A node with slack 1 can have its depth increased by 1 level without affecting the overall depth of the DAG. The maximum slack, among all nodes in the network, is represented by $MaxSlack$. Using the slack and maximum slack, we define the slack weight of a node z to be:

$$SlackWeight(z) = 1 + \kappa \cdot \left[\frac{MaxSlack - Slack(z)}{MaxSlack} \right] \quad (5)$$

where κ is a positive real number. The definition of slack weight implies that nodes with 0 slack have a slack weight of $1 + \kappa$ and nodes with $MaxSlack$ have a slack weight of 1. We compute the slack values and weights of nodes upfront in the circuit's unmapped DAG. We have observed that a node's slack in the unmapped DAG generally correlates well with the slack of the node's covering LUT in the mapped network.

As mentioned in Section 2.1, for a cut, C_z , for node z , the replicated nodes, $RNodes(C_z)$, in $Nodes(C_z)$ are those nodes that fanout to a node outside of $Nodes(C_z)$ (other than z itself). This is illustrated in Figure 2, where $Nodes(C_z) = \{z, a, b\}$ and $RNodes(C_z) = \{a\}$. If $Nodes(C_z)$ is implemented as a LUT in the mapping solution, the logic function of node a must be replicated in a second LUT, since it has fanouts outside the first LUT. The formal definition of the replicated nodes in a cut C_z is:

$$RNodes(C_z) = \{v \in Nodes(C_z) \mid v \neq z, (\exists u \mid u \in output(v), u \notin Nodes(C_z))\} \quad (6)$$

We can now define the replication cost of a cut C_z :

$$RCost(C_z) = \frac{1}{SlackWeight(z)} \times \sum_{v \in RNodes(C_z)} \left[\left(\sum_{u \in input(v)} f_u \right) - \lambda \cdot f_v \cdot |output(v) \cap Nodes(C_z)| \right] \quad (7)$$

The first summation is over the replicated nodes. For each replicated node, v , we sum the activities of the signals driven by v 's fanins. Recall that in Section 2.1, we showed that replicating a node generally increases the fanout of its fanin nodes. The consequences of this on power consumption depends on the activities of the signals driven by these fanin nodes and hence, $RCost$ is increased in proportion to these activities. The second term in the square brackets is negative and its intent is similar to the second summation term in (4). By including a replicated node v in a LUT, we “capture” a subset of its fanout connections within the LUT. We reduce the $RCost$ in proportion to the product of the number of captured connections and activity of these captured connections. λ is a coefficient that we set to 0.5 in our experiments.

The intuition behind dividing by the slack weight in (7) is to reduce the replication cost for nodes whose depth in the mapping solution is likely to impact the overall depth of the mapped circuit. Nodes with low slack will have a slack weight that is substantially larger than one. For such “critical” nodes, it is more important that we select a best cut that optimizes depth rather than one that avoids logic replication. We achieve this by reducing the replication cost through dividing by the larger slack weight.

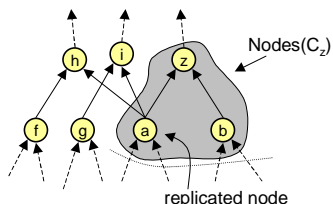


Figure 2. Identifying the replicated nodes

3.3. Mapping

The mapping phase of our algorithm is similar to that of FlowMap [4]. We initialize a FIFO queue to contain all of the primary output nodes. We remove a node v from the queue and then recall $C_v = BestCut(v)$. We implement the subnetwork corresponding to $Nodes(C_v)$ as a LUT in the mapping solution. Each node in $Support(C_v)$ is then added to the end of the FIFO queue (if not already in the queue). The process of removing nodes from the queue, using their best cuts to establish LUTs in the mapping solution, and adding the support of these cuts to the end of the queue continues until the queue contains only primary inputs and we have fully mapped the network into LUTs.

4. Experimental Study and Results

Our algorithm has been implemented in the C language within the Berkeley SIS framework [14]. For our experiments, we use 29 of the largest MCNC

combinational circuits (each uses > 300 LUTs). Prior to technology mapping, each benchmark circuit was optimized in SIS using *script.rugged* [14] and then transformed into a network of 2-bounded functions using *dmig* [15].

We compare our technology mapper with two publicly available technology mappers: 1) FlowMap [4], which maps circuits in a depth-optimal manner and 2) FlowMap-r [5], which optimizes both depth and area by relaxing the depth optimality on portions of a circuit that are not depth-critical and then performing duplication-free mapping. Additionally, we consider the effect of using various area-reducing post-processing routines, including FlowPack (FP) [4], MP-Pack (MP)¹ [15].

For our algorithm, we set parameters α and β in (2) to be 1 and 0.0001, respectively, reflecting a preference for optimizing depth over power. We chose the value of parameter γ (the replication cost weight) individually for each circuit such that power was minimized while meeting certain depth constraints (described below). To compute the value of γ for each circuit, we used an iterative approach in which γ was initially set to a small value and then increased gradually. For each value of γ considered, we invoked steps 2 and 3 of our algorithm (costing and mapping), keeping track of the mapping solution with the best power characteristics. In practice, a binary search could be used to select the best value for γ , at a small reduction in quality. We follow our algorithm by calling MP-Pack as a post-processing routine.

To estimate power consumption using (1), we require the capacitance of each net. Actual net capacitance is not known until layout is complete. We therefore estimate capacitance using structural properties of the circuit. We developed our capacitance model by using VPR [16] to place and route our benchmark circuits, mapped using the FlowMap-r algorithm. The logic block in the FPGA architecture we targeted contained a cluster of four 4-LUT/flip-flop pairs. The routing network was comprised of wire segments of length 4 (span 4 logic blocks), with half of the routing switches being buffered and half unbuffered. This FPGA architecture has been shown to be efficient from both the area and delay perspective [17]. We used VPR’s built-in interconnect model, with resistance and capacitance values based on a 0.18 μ m TSMC process [18]. Following the placement and routing of each circuit, we extracted capacitance data for each net (incl. metal and transistor capacitance). In generating our model, we considered only the routing of nets between logic blocks, ignoring the connections within a logic block. The results of this analysis are shown in Figure 3. The horizontal axis represents the number of pins per net; the

¹ MP-Pack was executed with node duplication off.

vertical axis shows total net capacitance. Each point in the figure represents the capacitance of a single net in one of our benchmarks. In addition to the raw data, the figure shows a line of best fit, which has the following equation:

$$C_i = 1.05 + 1.55 \cdot \text{num_pins}(net_i) \quad (8)$$

where $\text{num_pins}(net_i)$ represents the total number of pins on net_i . We use (8) in computing our power results. However, as is evident in Figure 3, a net with a given number of pins can have a range of capacitance values. Therefore, an important direction for future work is to further validate our results by placing and routing the mapped circuits using power-aware layout tools.

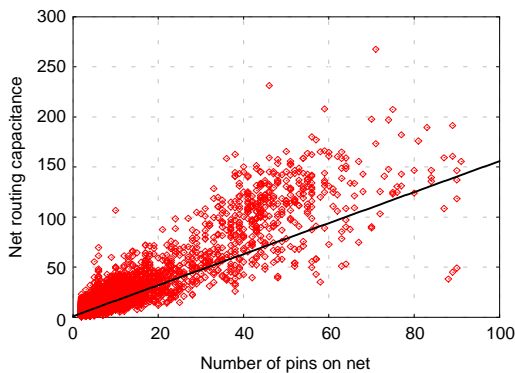


Figure 3. Routing capacitance vs. # of net pins

To measure power using (1), we also need an activity value for each net. We compute this value using the power characterization capabilities that are built-in to SIS. Specifically, for each primary input, i , SIS allows one to specify a signal probability, p_i , which is the probability that the value at the primary input is logic ‘1’ during circuit operation. SIS propagates primary input signal probabilities through the network to yield a probability for each internal node. The activity value for the net driven by an internal node, n , is computed by SIS using $f_n = 2 \cdot p_n \cdot (1 - p_n)$ [7]. For most of the results in this paper, we set the signal probability of all primary inputs to be 0.5, corresponding to a primary input activity of $2 \cdot (0.5) \cdot (1 - 0.5) = 0.5$. For a limited set of results, we investigate the effect of this choice by re-computing the power of already-mapped solutions using randomly chosen primary input signal probabilities.

4.1. Results

We first consider mapping circuits into 4-LUTs in a depth-optimal manner. Figure 4 shows the average power, area (# of LUTs) and number of connections for this case. The power for a circuit was computed by first estimating

the capacitance of each net (between LUTs) using (8); total circuit power was then computed using (1). Figure 4 includes the number of connections because we believe this metric correlates with overall routing complexity, as it represents the total number of net load pins to be routed. The numbers in Figure 4 were computed by first determining the increase in power, area and number of connections for each mapped circuit in comparison with the mapping solution produced by our algorithm. These increases were then averaged across all circuits for a given mapping approach.

Figure 4 shows that the mapping solutions produced by our approach have substantially less power dissipation than those produced by other approaches. The method most competitive with ours is FlowMap-r followed by MP-Pack; the solutions produced using this technique require 14.2% more power than ours, on average. Table 1 shows detailed results comparing our algorithm with FlowMap-r + MP-Pack for a subset of the 29 circuits. We observe that the gains offered by our algorithm are consistent; specifically, our power dissipation was equal to or better than FlowMap-r + MP-Pack for 27 of 29 circuits. Further, our algorithm also improves the area and number of connections slightly (~5%); hence, we believe our results will remain valid after layout.

Figure 4 shows that for optimal depth 4-LUT mapping solutions, power can vary by as much as 40% on average, depending on the mapping approach used. This variation is much higher than that of area or the number of connections, which can change by about 25% and 20%, respectively. Thus, we conclude that simply knowing that a circuit has been mapped in a depth-optimal manner does not allow one to make inferences regarding power.

Another interesting feature of Figure 4 is that it shows the effect of the post-processing routines. We see that FlowPack (FP) reduces the number of LUTs in mapping solutions substantially; however, it increases the power as well as the number of connections to route. FlowPack is a variation on FlowMap that maximizes cut volume; that is, it maximizes the number of nodes that are “packed” into each LUT, permitting logic replication. The logic replication performed by FlowPack may lead to increased net fanout and higher power. On the other hand, the data in Figure 4 show that MP-Pack is about as effective as FlowPack in reducing area and also reduces both power and the number of connections to route.

Figure 5 shows how power varies when the optimal depth constraint is relaxed and circuits are mapped with optimal depth + 1. Results are given for our algorithm as well as FlowMap-r + MP-Pack for 4-LUTs and larger, 5-LUTs. Again, the numbers in the figure are normalized to the 4-LUT, depth-optimal solution produced by our algorithm. As depth is increased, we see that greater amounts of logic replication can be eliminated, which

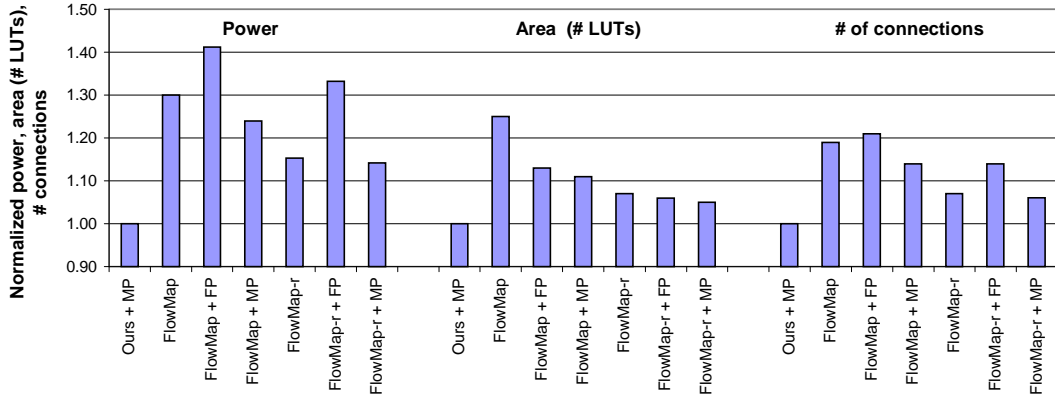


Figure 4. Power, area, number of connections in depth-optimal 4-LUT mapping solutions

permits further reductions in power. Specifically, relaxing the depth constraint by one level allows our algorithm to reduce power by about 8% over the depth-optimal case for 4-LUTs and 10% for 5-LUTs. Note that the 4-LUT mapping solution produced by FlowMap-r + MP-Pack with relaxed depth uses more power than the depth-optimal solution produced by our approach.

Table 1. Detailed results for depth-optimal 4-LUT mapping solutions

Circuit	Depth	Power increase FlowMap-r + MP versus ours + MP	Area increase FlowMap-r + MP versus ours + MP	# conns increase FlowMap-r + MP versus ours + MP
C3540	15	1.11	1.09	1.05
C5315	11	1.02	1.03	1.04
alu4	7	1.20	1.11	1.11
apex1	7	1.14	1.00	1.00
apex2	8	1.15	1.08	1.07
apex3	6	1.19	1.10	1.09
ex5p	7	1.26	0.97	0.97
apex5	5	1.05	1.05	1.03
cordic	9	1.08	1.04	1.06
cps	5	1.16	1.06	1.05
dalu	6	1.16	1.03	1.07
des	7	1.06	1.01	1.02
Average for these circuits		1.13	1.05	1.05
Average across 29 circuits		1.14	1.05	1.06

Modern commercial FPGAs contain 4-input LUTs; however, some devices allow two 4-LUTs to be combined into a 5-LUT [1]. Hence, mapping to 5-LUTs is also an important problem. The results in Figure 5 show that for FlowMap-r + MP-Pack, 5-LUT mapping solutions actually require more power than the 4-LUT solutions; whereas, for our algorithm, the 5-LUT solutions require slightly less power than the 4-LUT solutions. The depth of the 5-LUT mapping solutions is generally smaller than the depth of the 4-LUT mapping solutions - it appears to be more expensive from the power viewpoint for FlowMap-r to achieve this smaller depth. Figure 5 shows that the improvements offered by our algorithm over FlowMap-r are larger for 5-LUTs than 4-LUTs. The larger LUTs can

cover a larger portion of the input network and thus appear to offer more potential for power optimization. For one of the circuits, *spla*, the depth-optimal 5-LUT solution produced by FlowMap-r + MP-Pack used 7 times more power than the solution produced by our algorithm. Since this circuit affected the average substantially, we removed it from the data presented in Figure 5. If this circuit is included, the average power of the 5-LUT mapping solutions of FlowMap-r + MP-Pack increases to nearly 1.5 for the depth optimal case and 1.16 for the relaxed depth case.

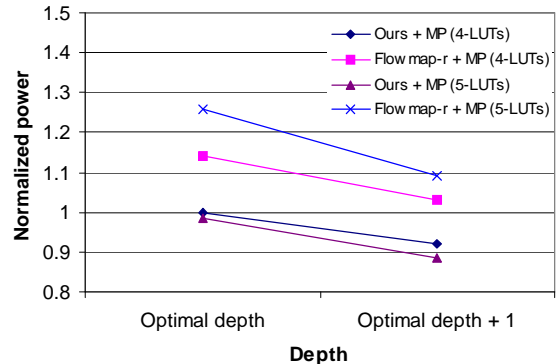


Figure 5: Power results for other depths, 5-LUTs

The power results presented above were computed based on the primary inputs of each circuit being set to have identical switching activities (0.5). A problem that arises frequently in low-power synthesis is that input switching activities are not known at synthesis time, or the set of switching activities used during synthesis do not reflect the stimulus applied to a circuit in actual field operation. To investigate the dependence of our results on switching activities used during technology mapping, we re-computed the power for the already mapped circuits using randomly chosen input switching activities. Specifically, we set the signal probability for each primary input to a random number between 0.1 and 0.9, corresponding to a randomly chosen switching activity

between 0.18 and 0.5. We re-computed the power of the depth-optimal 4-LUT mapping solutions produced by our algorithm as well as FlowMap-r + MP-Pack. The results showed that the improvements offered by our algorithm over FlowMap-r + MP-Pack degraded by only 1-2% on average. Thus, it appears that our improvements to power remain considerable, even when switching activities deviate from those used during technology mapping.

Unfortunately, we were not able to compare our algorithm directly with PowerMap [9], which, like our algorithm, optimizes depth and power. However, the results in that paper compare PowerMap with FlowMap for LUTs with 5-inputs, showing a power improvement of about 15%. The results in Figures 4 and 5 show our approach provides gains over FlowMap that exceed this margin.

5. Conclusions and Future Work

In this paper, we presented a new algorithm for power-aware mapping that allows one to trade-off depth and power. One novel aspect of our approach is that it takes an activity-aware approach to logic replication, which has been shown to significantly affect the power of technology mapped circuits. In an experimental study, we showed that our algorithm produces solutions that require less power than competing techniques. We also showed that for a given depth of mapping solution, circuit power can vary considerably, depending on the technology mapping approach used and the choice of area-reducing post-processing routine. We observed that power reductions are possible when the requirement of depth optimality is relaxed.

One direction for future work is to further validate our results by placing and routing the mapping solutions with power-aware placement and routing tools, allowing actual capacitance values to be used when computing power. A second area for future work is to simultaneously consider the trade-offs between depth, area and power in LUT-based technology mapping.

6. Acknowledgments

The authors thank Andy Ye for his help with extracting routing data from VPR, Jason Cong (UCLA) for providing the RASP package, and Vaughn Betz and Jonathan Rose for supplying VPR. The authors gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada.

7. References

[1] *Virtex II Platform FPGA Data Sheet*, Xilinx Inc., 2002, <http://www.xilinx.com/apps/virtexapp.htm>.

- [2] V. George and J. Rabaey, *Low-Energy FPGAs: Architecture and Design*, Kluwer Academic Publishers, Boston, 2001.
- [3] R.J Francis, J. Rose, Z. Vranesic, "Chortle-crf: Fast Technology Mapping for Lookup Table-Based FPGAs," *ACM/IEEE Design Automation Conf.*, 1991, pp. 227 - 233.
- [4] J. Cong and Y. Ding, "FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs," *IEEE Trans. on CAD*, Vol. 13, No. 1, 1994, pp. 1 - 12.
- [5] J. Cong and Y. Ding, "On Area/Depth Trade-off in LUT-Based FPGA Technology Mapping," *IEEE Trans. on VLSI Systems*, Vol. 2, No. 2, 1994, pp. 137 - 148.
- [6] L. Shang, A. Kaviani, K. Bathala, "Dynamic Power Consumption Virtex-II FPGA Family," *ACM Int. Symp. on FPGAs*, 2002, pp. 157 - 164.
- [7] G. Yeap, *Practical Low Power Digital VLSI Design*, Kluwer Academic Publishers, Boston, 1998.
- [8] A. H. Farrahi and J. Sarrafzadeh, "FPGA Technology Mapping for Power Minimization," *Int. Workshop on Field-Programmable Logic and Applications*, 1994, pp. 66 - 77.
- [9] H. Li, W-K. Mak and Srinivas Katkoori, "LUT-Based FPGA Technology Mapping for Power Minimization with Optimal Depth," *IEEE Computer Society Workshop on VLSI*, 2001, pp. 123 - 128.
- [10] Z-H. Hong Wang, E-C. Liu, J. Lai and T-C. Wang, "Power Minimization in LUT-Based FPGA Technology Mapping," *ACM/IEEE Asia South Pacific Design Automation Conf.*, 2001, pp. 635 - 640.
- [11] M. Nemani and F. Najm, "Towards a High-Level Power Estimation Capability," *IEEE Trans. on CAD*, Vol. 15, No. 6, 1996, pp. 588 - 598.
- [12] J. Cong, C. Wu and E. Ding, "Cut Ranking and Pruning: Enabling A General And Efficient FPGA Mapping Solution," *ACM Int. Symp. on FPGAs*, 1999, pp. 29 - 35.
- [13] M. Schlag, J. Kong and P.K. Chan, "Routability-Driven Technology Mapping for Lookup Table-Based FPGAs," *IEEE Trans. on CAD*, Vol. 13, No. 1, 1994, pp. 13 - 26.
- [14] E.M. Sentovich et al., "SIS: A System for Sequential Circuit Synthesis," *UC Berkeley, Memorandum No. UCB/ERL M92/41*, Electronics Research Laboratory, May 1992.
- [15] K.C. Chen et al., "DAG-Map: Graph-Based FPGA Technology Mapping for Delay Optimization," *IEEE Design and Test of Computers*, September 1992, pp. 7 - 20.
- [16] V. Betz and J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research," *Int. Workshop on Field-Programmable Logic and Applications*, 1997, pp. 213 - 222.
- [17] V. Betz, J. Rose and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*, Kluwer Academic Publishers, Boston, 1999.
- [18] *TSMC 0.18 μ m process*, TSMC Corp., 2002, <http://www.tsmc.com/english/technology/t0103.htm>.