ON THE INTERACTION BETWEEN POWER-AWARE FPGA CAD ALGORITHMS

Julien Lamoureux and Steven J.E Wilton

Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, B.C., Canada

ABSTRACT

As Field-Programmable Gate Array (FPGA) power consumption continues to increase, lower power FPGA circuitry, architectures, and Computer-Aided Design (CAD) tools need to be developed. Before designing low-power FPGA circuitry, architectures, or CAD tools, we must first determine where the biggest savings (in terms of energy dissipation) are to be made and whether these savings are cumulative. In this paper, we focus on FPGA CAD tools. Specifically, we describe a new power-aware CAD flow for FPGAs that was developed to answer the above questions.

Estimating energy using very detailed post-route power and delay models, we determine the energy savings obtained by our power-aware technology mapping, clustering, placement, and routing algorithms and investigate how the savings behave when the algorithms are applied concurrently. The individual savings of the power-aware technology-mapping, clustering, placement, and routing algorithms were 7.6%, 12.6%, 3.0%, and 2.6% respectively. The majority of the overall savings were achieved during the technology mapping and clustering stages of the power-aware FPGA CAD flow. In addition, the savings were mostly cumulative when the individual power-aware CAD algorithms were applied concurrently with an overall energy reduction of 22.6%.

1. INTRODUCTION

Power consumption has become a critical concern in the semiconductor industry. As the heat generated by integrated circuits begins to exceed the ability of packaging to dissipate this heat, designers are forced to sacrifice performance in order to meet power budgets. Furthermore, the increased demand for low-power integrated circuits for hand-held applications provides additional incentive for the development of new techniques to reduce power consumption. Power consumption is especially critical in Field-Programmable Gate Arrays (FPGAs). An FPGA's programmability is afforded through the use of long routing tracks and programmable switches. These switches are laden with parasitic capacitance. During high-speed operation, the switching of these tracks causes significant power dissipation. Already, many FPGA vendors report that power dissipation is one of the primary concerns of their customers.

FPGA power consumption can be reduced by optimizing at the circuit level, architecture level, and at the Computer-Aided Design (CAD) level. There have been several low-power architectures and CAD tools described by previous researchers [1,7,12,14,15]. However, these have all been "point-solutions", in that each considers only a single CAD algorithm or architecture. In this paper, we describe a suite of power-aware CAD algorithms, and use this suite to answer two questions:

1. What stages of the FPGA CAD flow are most suited to power minimization? In this paper, we focus on technology mapping, clustering, placement and routing. We expect that high-level synthesis algorithms would also be amenable to reducing power, but we have not yet investigated this.
2. Are the power savings from individual power-aware stages cumulative? In other words, we wish to know whether savings at one stage can impact the savings that can be achieved in subsequent stages.

Thus, the primary goal will be to understand the interaction between the power reduction techniques in each stage of the CAD flow. Only by understanding where energy savings can be expected, and how these savings interact, can we expect to make significant progress in creating low-power FPGA CAD tools.

This paper is organized as follows. Section 2 describes the experimental framework used to evaluate the performance of the power-aware CAD algorithms. Section 3 introduces terminology used throughout this paper. Sections 4 to 7 describe the new power-aware technology mapping, clustering, placing, and routing algorithms, and their results. Section 8 then combines the results from the individual stages of the flow to determine if the savings are cumulative. Finally, Section 9 presents our conclusions.

2. EXPERIMENTAL FRAMEWORK

To answer questions 1 and 2, we begin with a baseline FPGA CAD flow consisting of well-established algorithms, as shown in Figure 1. The baseline CAD flow consists of CutMap [5], T-VPACK [8], and VPR [2,9]. These algorithms are representative of algorithms used in commercial FPGA CAD flows.

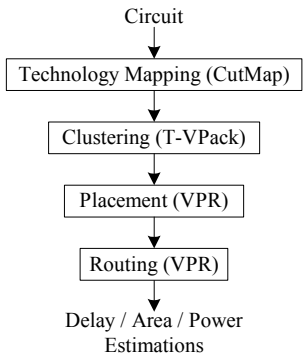

Figure 1. Baseline FPGA CAD flow.

To investigate the influence of each CAD stage on energy reduction, we replace each CAD stage with a power-aware algorithm. Initially, we replace only one CAD stage at a time, so

that we can examine and compare the impact of each stage on the energy reduction. Then, we replace multiple baseline stages with power-aware algorithms to investigate the interaction between the power-aware stages. In all cases, the power-aware algorithms we use are representative of power-aware algorithms that have either been published in the literature or are straightforward extensions of the baseline CAD algorithms.

To measure the effectiveness of our algorithms, we use detailed power and delay models. Regardless of which stage we are enhancing (to make it power-aware), we estimate the speed and power of our implementations after routing has been performed. This provides for much more accurate estimates than would be possible during higher-levels of the CAD flow, since only after routing can we accurately determine the resistance and capacitance associated with each net in the circuit.

The delay model used was that from VPR [2]. VPR models an FPGA at a low-level, taking into account specific switch patterns, wire lengths, and transistor sizes. Once a circuit is routed onto the FPGA, VPR extracts the resistance and capacitance information for each net, and uses the Elmore delay to produce delay estimates.

The power model is described in [12]. This model uses the same resistance and capacitance information as the delay model, and uses this information to estimate dynamic power, short-circuit power, and leakage power. Switching activity estimations for each wire of a circuit, which are required for calculating dynamic and short-circuit power, are computed using the transition density model along with a filter function used to emulate the effect of inertial delays of logic gates [11]. Although simulation-based activity estimation techniques provide more accurate estimates, the transition density technique is far faster, and in [11] was shown to work well.

The metric we use for comparing algorithms is the power-delay product or energy. When comparing power-aware FPGA CAD algorithms it is important to consider both power and delay. Using CAD algorithms to minimizing power at the expense of delay is ineffective since similar results can be achieved by simply slowing the system clock. For each experiment we use 20 large MCNC benchmark circuits. Each benchmark was optimized in SIS using script.rugged [13] and then transformed into a network of 2-input gates using dmig [3]. All experiments target island-style FPGAs implemented in a 0.18µm TSMC CMOS process.

3. TERMINOLOGY

Before presenting our results we review some terminology defined in [4,5] to describe Boolean networks. A Boolean network can be represented by a directed acyclic graph (DAG), where gates are represented by nodes and wires are represented by directed edges. Given a network $N=(V(N), E(N))$ with a source s and a sink t, a cut $(X, \bar{X})$ is a partition of the nodes in $V(N)$ such that $s \in X$ and $t \in \bar{X}$. The *cut-size* is the number nodes in X that are adjacent to some node in $\bar{X}$. A cut is *K-feasible* if its cut-size is smaller or equal to K. The set of nodes which are fanins of node v is denoted $input(v)$ and the set of nodes which are fanouts of node v denoted is $output(v)$. Given a subgraph H of the Boolean network, $input(H)$ denotes the set of distinct nodes outside H which supply input to the gates in H. A Boolean network is *K-bounded* if $|input(v)| \leq K$ for each node v in the network. The *depth* of a node v is the length of the longest path from any primary input of the network to v. Given a K-bounded network N, let N_v denote the subnetwork consisting of node v and all the predecessors of v. The label of v, denoted $label(v)$, is defined as the depth of the optimal K-LUT mapping solution of N_v. Finally, a node v that is an input to a K-LUT or to a primary output is denoted $rooted(v)$.

4. TECHNOLOGY MAPPING

The first stage of the FPGA CAD flow that we consider is technology mapping. Technology mapping transforms a netlist of gates and registers into a netlist of K-input lookup tables (K-LUTs) and registers. LUT-based technology mapping has been well studied [1,4,5,6,7]. The goal in this section is to understand how much of an influence a power-aware technology mapper can have in reducing the overall energy. To make our results concrete, we have implemented a power-aware technology mapper that uses techniques described in previous works [1,5,6].

Existing power-aware technology mapping algorithms typically reduce power by minimizing the switching activity of the wires between LUTs. In FPGAs, these wires are implemented using routing tracks with significant capacitance; charging and discharging this capacitance consumes a significant amount of power. Intuitively, by minimizing the capacitance of high activity wires, the total power of the final implementation may be reduced. The capacitance of high activity wires between LUTs can be minimized during technology mapping by implementing LUTs that encapsulate high activity wires, thereby removing them from the netlist.

Another power reduction technique, recently described in [1], is to minimize the number of wires between LUTs. This can be achieved by minimizing node duplication. Technology mappers that are not power-aware use node duplication to optimize for depth. However, this technique tends to increase the number of nodes and connections in an implementation, which increases the amount of power used by the implementation. To demonstrate this we compare FlowMap and CutMap. Both algorithms produce depth optimal solutions. However, CutMap also attempts to minimize area by avoiding unnecessary node duplication. The results are shown in Table 1. The 4-LUT circuits mapped using FlowMap have 12.6% more 4-LUTs, 7.7% more connections, and correspondingly dissipate 9.3% more energy than circuits mapped using CutMap.

4.1 Power-Aware Algorithm

Our power-aware algorithm, called Emap, incorporates both techniques described above. The EMap algorithm has three phases (see Figure 2).

The first phase of the algorithm begins by constructing the set of all K-feasible cuts for each node in the network using the technique outlined in [6]. The nodes are processed in topological order (beginning from the primary inputs) thereby guaranteeing that every node is processed after all of its predecessors. After all the cuts are found, each node is labeled with the depth that it would have in an optimal depth K-LUT mapping solution. These labels are needed during the second phase of the algorithm to determine the slack of each node. The slack is used to guide the algorithm and produce a network with optimal depth.

The second phase of the algorithm evaluates the cuts of each node in the network in reverse topological order (beginning from the

primary outputs). For each node, it chooses the cut with the lowest cost from one of two possible cut sets. If the node has no slack, only cuts that produce a depth-optimal mapping solution are considered; however, if it does have slack, all *K*-feasible cuts are considered. After selecting a cut, the nodes that fan into the cut are labeled as root nodes and their slack is updated.

The third and final phase of the algorithm generates the final *K*-LUT network by traversing the graph in reverse topological order and collapsing each node based on the cuts selected during the second phase.

```
/* Phase 1 */
foreach node v ∈ N do
   enumerate_K_feasible_cuts(v, K);
foreach node v∈ N do
   label(v) = compute_label(v);
   if (v ∈ primary_input(N) || v ∈ primary_output(N))
      rooted(v) = TRUE;
   else
      rooted(v) = FALSE;
end for
D_opt = max({label(v) | v ∈ N})
foreach node v∈ N do
   latest(v) = D_opt;
   slack(v) = latest(v) – label(v);
end for

/* Phase 2 */
foreach node v ∈ N do
   if (rooted(v) == TRUE)
      if slack(v) > 0
         (X_v, X̄_v) = choose_cut(K-feasible_cut(v));
      else
         (X_v, X̄_v) = choose_cut(min_height_K-feasible_cut(v));
      foreach u ∈ input (X_v, X̄_v) do
         rooted(u) = TRUE;
         latest(u) = min(latest(u), latest(v) – 1);
         slack(u) = latest(u) – label(u);
      end for
   end if
end for

/* Phase 3 */
form_LUT_network(N);
```

Figure 2. Pseudo-code of the EMap algorithm.

4.2 The Cost Function

During the second phase of the algorithm, the cut with the lowest cost is selected from the cutset of each node. The function used to determine the cost of each cut (X_v, $\overline{X_v}$), is:

$$\text{cost}(X_v, \overline{X_v}) = \frac{1+|\,rooted(\overline{X_v})\,|}{1+|\,\overline{X_v}\,|-|\,rooted(\overline{X_v})\,|} \bullet \sum_{u\in \text{input}(\overline{X_v})} \frac{weight(u)\cdot(1+\lambda\cdot act(u))}{|\,output(u)\,|}$$

where $\overline{X_v}$ is the set of nodes encapsulated within the LUT that corresponds to cut (X_v, $\overline{X_v}$), *rooted*($\overline{X_v}$) is the set of nodes in $\overline{X_v}$ that have been labeled as root nodes, *weight*(*u*) is 0 if node *u* has been (or is likely to be) labeled as a root node of a LUT and is 1 otherwise (to be explained below), *act*(*u*) is the estimated switching activity of the net driven by node *u*, λ is a constant that controls the relative importance of the activity factor, and *output*(*u*) is the set of nodes that are fanouts of node *u*.

The first part of the cost function is a quotient. Intuitively, the numerator of the quotient penalizes node duplications by increasing the cost of cuts that encapsulate nodes that have already been labeled as root nodes. The denominator, however, rewards cuts that encapsulate many nodes that have not been labeled as root nodes. Both help to minimize the number of LUTs and connections in the final solution.

The second part of the cost function is a summation over all the inputs nodes of $\overline{X_v}$. The numerator of the sum is the *weight-activity* product and the denominator is the *fanout size* of input node *u*. The *weight* factor minimizes node duplication by favoring cuts that reuse nodes that have already been cut, or that are likely to be cut in the future. The *activity* factor minimizes the switching activity of the connections by favoring cuts with lower input activities. The *fanout size* factor rewards cuts that have high-fanout input nodes. High-fanout nodes are difficult to encapsulate entirely; attempting to encapsulate them results in unnecessary node duplication. This is avoided by choosing high-fanout nets as root nodes. Finally, the summation implicitly favors cuts with fewer inputs since the cuts with fewer inputs tend to have lower sums.

Using this cost function, nodes with large fanouts are likely to be chosen as root nodes. To enhance the algorithms ability to minimize node duplication, the *weight* of nodes with large fanouts (3 or more) are set to 0 prior to phase 2. This gives cuts with high fanout nets a lower cost.

4.3 Technology Mapping Results

To evaluate the influence of the technology-mapper on the power consumption of the final circuit implementation, we use the CAD flow described in Section 2 with CutMap replaced by our power-aware mapper. It is important to note that we take each circuit through the entire flow, and estimate power and delay after routing. This is different than in previous works [1,7] where the reduction in average switching activity is used to evaluate power-aware technology mapping. In our case, since we wish to compare these improvements to those obtained in later CAD stages, we need to obtain post-route power and delay estimates.

As shown in the last column of Table 1, the energy reduction, averaged over all benchmark circuits, is 7.6%, 8.4%, and 8.2% for LUT sizes of 4, 5, and 6 respectively. In some previous works, improvements of up to 17% have been reported; however, in these works, either a simplified power model (obtained before placement and routing and that only consider dynamic routing power) was employed, or else comparisons were made to FlowMap or another similar technology mapper. Comparing our results to FlowMap using the simplified model described in [7], our improvement is approximately 21%.

Table 1. Technology Mapping Results.

LUT Size	Algorithm	Nodes		Connections		Energy (nJ)	
		Mean	% Diff	Mean	% Diff	Mean	%Diff
4	FlowMap	2900	12.6	11576	7.7	2.39	9.3
	CutMap	2576	0	10746	0	2.18	0
	EMap	2441	-5.2	9705	-9.7	2.01	-7.6
5	FlowMap	2554	18.5	11301	11.9	2.53	11.9
	CutMap	2156	0	10102	0	2.26	0
	EMap	2079	-3.6	9102	-9.9	2.07	-8.4
6	FlowMap	2109	18.4	10179	11.6	2.59	12.1
	CutMap	1782	0	9118	0	2.31	0
	EMap	1771	-0.6	8331	-8.6	2.12	-8.2

The improvements of the technology mapper come primarily from the minimization of node duplication. The switching activity improvements account for only a small fraction of the energy savings. As we increase the relative importance of the switching activity factor, λ, the average switching activity of the wires between the LUTs decreases; however, node duplication increases. The resulting increases in the number of nodes and the number of connections more than counteract the benefits of the activity reduction. The best results, shown in Table 1, where obtained when λ is set to 0.25.

5. CLUSTERING

Modern island-style FPGAs have clustered logic blocks which consist of multiple LUT/register pairs called logic elements (LEs). The clusters (LABs in Altera parts and CLBs in Xilinx parts) typically have between 4 and 10 LEs that are locally interconnected. Connections within the logic blocks are faster and dissipate less energy than connections between logic blocks.

Clustering algorithms are used to pack the LUTs and registers into clusters. Traditional clustering goals include minimizing area, minimizing delay, and maximizing routability. To minimize area, clustering algorithms try to pack clusters to full capacity in order to minimize the number of clusters. Delay is minimized by packing LUTs that are on a critical-path together in order to exploit local routing, which is faster than global routing. Finally, routability is improved by minimizing the number of inputs used by each cluster.

Intuitively, we would expect clustering to be more effective than technology mapping at reducing power, since clusters are typically larger (commercial parts contain as many as 10 LUTs per cluster). On the other hand, encapsulating high activity nodes into clusters does not eliminate these nodes entirely, as it does in technology mapping. An interconnection between LUTs within a cluster still requires a connection; however, the capacitance of this intra-cluster connection is much smaller than the capacitance of the inter-cluster connections.

5.1 Clustering Algorithm

To investigate these tradeoffs, we have extended the T-VPack algorithm from [8] by modifying the cost function. In T-VPack, LUTs are packed one at a time. For each LUT, an attraction function is used to select a seed LUT from the set of all LUTs that have not already been packed. After a seed LUT is packed into the new logic block, new LUTs are selected using a second attraction function. LUTs are packed into the cluster until it reaches full capacity or all cluster inputs have been used. If all the cluster inputs become occupied before the cluster reaches full capacity, a hill-climbing technique is applied which looks for LUTs that do not increase the number of inputs used by the cluster.

Of particular interest are the two attraction functions. The seed attraction function is used to select the initial LUT to pack into a new cluster. In the original algorithm, one of the LUTs on the most time-critical path is chosen. The second attraction function selects a LUT to be packed into a partially filled cluster. In the original algorithm, this second attraction function is (for a LUT B being considered for cluster C):

$$Attraction(B) = \alpha \cdot Crit(B) + (1-\alpha)\frac{|\, Nets(B) \cap Nets(C)\,|}{G}$$

where *Crit(B)* is a measure of how close LUT B is to being on the critical path, *Nets(B)* is the set of nets connected to LUT B, *Nets(C)* is the set of nets connected to the LUTs already selected for cluster C, and G is a normalizing factor. The first term in this attraction function gives priority to nodes on the critical path. The second term gives priority to LUTs that share nets with the LUTs already packed into the cluster.

In our power-aware clustering algorithm, we modify these attraction functions. The first attraction function is modified so as to select a LUT whose input and output wires have the highest switching activity.

The second attraction function, which selects the remaining LEs that are packed into each cluster, is modified as follows (for an LE B being considered for cluster C):

$$Attraction(B) = \alpha \cdot Crit(B) + (1-\alpha)\cdot\left[\begin{array}{l}(1-\beta)\cdot\dfrac{\sum Weight(i)\,|\, i \in Nets(B)\cap Nets(C)}{G} + \\ \beta\cdot\dfrac{\sum Activity(i)\,|\, i \in Nets(B)\cap Nets(C)}{G\cdot Activity_{Avg}}\end{array}\right]$$

where *Crit(B)* is a measure of how close LE B is to being on the critical path, *Nets(B)* is the set of nets connected to LE B, *Nets(C)* is the set of nets connected to the LEs already selected for cluster C, *Activity*(i) is the estimated switching activity of net i, $Activity_{\text{Avg}}$ is the average switching activity of all the nets in the user circuit, α and β are user-defined constants which determine the relative importance of each attraction component, and G is a normalizing factor.

The first term of the new attraction function is the same as before, the second is modified, and the third is new. Instead of measuring the cardinality of the set of shared nets for each LE, the second term sums the *weight* of each shared net. The weight of a net is 1 for most nets; however, the weight is 2 for nets that are likely to be fully encapsulated into the current cluster. A weight of 2 is assigned to nets that are small (fewer than 4 pins) and that have not already been connected to any other cluster. The weight factor increases the probability of encapsulating nets entirely within a cluster by favoring nets that are more easily encapsulated. The third term of the attraction function minimizes the switching activity of connections between logic blocks by attracting high activity nets inside the logic blocks. The term favors LEs that

share high activity nets with the LEs that are already packed in the current logic block. Values of 0.0 for α and 0.6 for β were found experimentally to produce the most energy-efficient results.

5.2 Clustering Results

Figure 3 compares the energy dissipation of circuits clustered using T-VPack and the new power-aware clusterer (P-T-VPack). In both cases, the baseline technology mapper, placer, and router were used. Again, the energy results are obtained after routing. The graph illustrates that the energy minimization becomes more effective as the cluster size is increased. Larger clusters encapsulate more wires allowing the algorithm to remove more high-activity wires from global routing. For clusters of size 4, the energy is reduced by 12.6%.

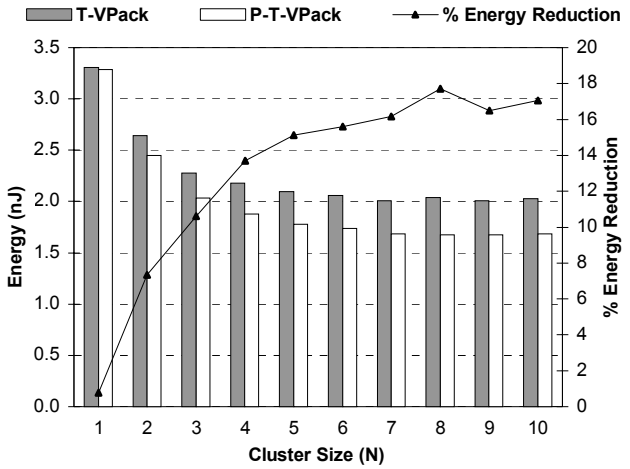

Figure 3. Power-aware clustering results versus cluster size.

In Table 2 we examine the energy reductions in more detail. For clusters with four LEs, the power-aware clusterer reduces the number of inter-cluster connections by 1.0% and the average inter-cluster switching activity by 20.8%. In contrast with the technology mapper, the improvements from the clustering algorithm come primarily from the minimization of switching activity.

Table 2. Clustering results.

	T-VPack	P-T-VPack	% Diff.
# Connections	6268	6206	-1.0
Average Activity	0.298	0.236	-20.8
Energy (nJ)	2.18	1.88	-12.6

6. PLACEMENT

After being packed, the clusters are mapped to physical locations on the FPGA. This stage of the CAD flow is called placement. Intuitively, a good placement can have a significant impact on power. If clusters connected by high-activity nets are placed near each other, these high-activity nets will likely be short, and thus, consume less power. On the other hand, unlike technology mapping, a placement algorithm can not eliminate high-activity nets all together; it can only make these nets shorter. In cases when there are many high-activity nets, it may not be possible to place all clusters connected by these nets close together. Similarly, in cases when there are timing-critical nets that also have low switching activity, the delay of the circuit may increase. This delay increase may counteract the power reduction, thereby reducing the overall energy savings. To investigate these tradeoffs, we modified an existing timing-aware placement algorithm to optimize for power using a technique similar to that described in [15].

6.1 Placement Algorithm

Our baseline algorithm was T-VPlace [9], a part of the VPR tool suite [2]. T-VPlace is based on simulated annealing. The algorithm starts with a random initial placement of the circuit on the FPGA, after which pairs of logic blocks are randomly selected and then swapped for a large number of iterations. Each swap is evaluated to determine if it should be kept or not. If the swap decreases the cost, as defined by a cost function, the swap is always kept; however, if the cost increases, the swap may or may not be kept. The probability of keeping a seemingly-bad swap decreases as the algorithm executes.

The cost function used by T-VPlace has two components. The first component is the sum of the bounding box dimensions of all nets. That is, if there are N_{nets} nets, and $bb_x(i)$ and $bb_y(i)$ are the x and y dimensions of the bounding box of net i, then:

$$Wiring\ Cost = \sum_{i=1}^{N_{nets}} q(i) \cdot [bb_x(i) + bb_y(i)]$$

The term $q(i)$ is used to scale the bounding boxes to better estimate wirelength for nets with more than 3 terminals, as described in [9]. The second component is used to evaluate the timing cost of a potential placement. The timing cost is:

$$Timing\ Cost = \sum_{\forall i,j \in circuit} Delay(i,j) \cdot Criticality(i,j)^{CE}$$

where $Delay(i,j)$ is the estimated delay of the connection from source i to sink j, CE is a constant, and $Criticality(i,j)$ is an indication of how close to the critical path the connection is [9]. The total cost is the sum of the wiring cost and timing cost for all nets:

$$\Delta C = \lambda \cdot \frac{\Delta Timing\ Cost}{Previous\ TimingCost} + (1-\lambda) \cdot \frac{\Delta Wiring\ Cost}{Previous\ WiringCost}$$

where $PreviousTimingCost$ and $PreviousWiringCost$ are auto-normalizing factors that are updated once every temperature, and λ is a user-defined constant which determines the relative importance of the cost components.

To make this cost function power-aware, a third component is added to T-VPlace's cost function. The power cost component estimates the power consumption of each net by multiplying their bounding box and switching activity:

$$Power\ Cost = \sum_{i=1}^{N_{nets}} q(i) \cdot [bb_x(i) + bb_y(i)] \cdot Activity(i)$$

Like the timing and wiring components, the power component of the cost function is auto-normalized with a $PreviousPowerCost$ factor, which is updated once every temperature. The relative

importance of the power component is controlled with a user-defined constant, γ.

6.2 Placement Results

The power-aware algorithm (called P-T-VPlace) produced marginal but consistent improvements in terms of energy when compared with T-VPlace. As shown in the fourth column of Table 4, the post-routing energy dissipation was reduced by 3.0%, where all of the 20 benchmarks showed improvement. Note again that these improvements were for the placement algorithm only; the baseline algorithms were used for the other three CAD stages.

Intuitively, the P-T-VPack algorithm attempts to place clusters connected with high-activity nets close to each other. To investigate to what extent this is happening, we examined the relationship between the switching activity and the capacitance of each net after routing. We divided the nets of each circuit into groups, based on their activities (the first group consisted of nets with an activity of 0.0 to 0.1, the second group consisted of nets with an activity of 0.1 to 0.2, etc.). For each net, we found the post-routing capacitance using the baseline and power-aware placement algorithms. The total capacitance of all the nets in each group was then summed, and the results were plotted in Figure 4. This plot shows that high-activity nets are more likely to have a low capacitance when the power-aware placement algorithm is used, compared to when the baseline placement algorithm is used.

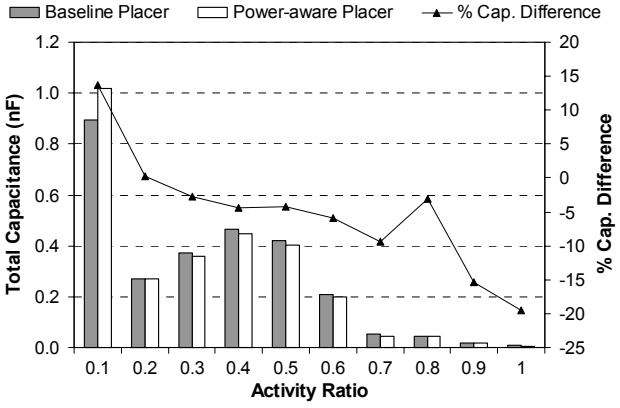

Figure 4. P-T-VPlace (wire cap. vs. switching activity).

Examining the results further, the power-aware placer reduces global routing power by 6.7% compared to the baseline placer. The critical-path delay, however, increases by 4.0%, thereby counteracting much of the power reductions. The delay increase is incurred when critical-path nets have low switching activity. When switching activity is not considered, all critical-path nets are kept short in order to reduce delay. However, when switching activity is considered, critical-path nets with low switching activity are not kept as short as before.

7. ROUTING

Once clusters are assigned to physical locations on the FPGA, connections between the clusters must be routed through the FPGA's prefabricated programmable routing fabric. Routing is more complex in FPGAs than in any other implementation medium, since only a limited number of programmable switches between routing tracks are provided. Intuitively, we would not expect a power-aware router to significantly impact the power of the final implementation. Although it is possible for a non-power-aware router to take a circuitous route for a high-activity net, experiments have shown that this rarely happens.

7.1 Routing Algorithm

To quantify this intuition, we have modified the router in VPR to be power-aware. The original router uses a negotiated congestion-delay algorithm based on PathFinder [10]. During initial iterations, an overuse of routing resources is allowed (in other words, it is acceptable for more than one net to share a routing wire). In later iterations, however, the penalty for this overuse is increased, until no wire is used by more than one net.

The baseline VPR router uses the following cost function to evaluate a routing track n while forming a connection from source i to sink j:

$$Cost(n) = Crit(i,j)\cdot delay(n) + (1 - Crit(i,j))\cdot b(n)\cdot h(n)\cdot p(n)$$

The cost function has a delay term and a congestion term. The delay term is the product of the Elmore delay of node n and $Crit(i,j)$ as defined in Section 6.1. The congestion term, which has more weight when the criticality is low, has three components: $b(n)$ is the "base cost", $h(n)$ is the historical congestion cost, and $p(n)$ is the present congestion of node n. The value of $p(n)$ is increased gradually as the algorithm progresses to discourage node sharing, allowing the algorithm to produce a legal solution.

To make the router power-aware, we modified the cost function as follows:

$$Cost(n) = Crit(i,j)\cdot delay(n) + (1 - Crit(i,j))\cdot [ActCrit(i)\cdot cap(n) + (1 - ActCrit(i))\cdot b(n)\cdot h(n)\cdot p(n)]$$

where $cap(n)$ is the capacitance associated with routing resource node n and $ActCrit(i)$ is the activity criticality of net i:

$$ActCrit(i) = \min\left(\frac{Activity(i)}{MaxActivity}, MaxActCrit\right)$$

where $Activity(i)$ is the switching activity in net i, $MaxActivity$ is the maximum switching activity of all the nets, and $MaxActCrit$ is the maximum activity criticality that any net can have. Setting $MaxActCrit$ to 0.99 prevents nets with very high activity from completely ignoring congestion.

The delay term is left unchanged; when criticality is high, the cost focuses on Elmore delay. The second term, however, is modified to consider the capacitance of a routing resource node when the activity of the net is high.

7.2 Routing Results

To investigate the effectiveness of the router on reducing energy, we used the new router with the baseline algorithms for all other CAD stages. The improvements, shown in the third column of Table 4, were similar to those achieved during placement. The average post-routing energy disspation was reduced by 2.6%. All of the 20 benchmarks showed an improvement.

Intuitively, this algorithm attempts to route high-activity nets using routing resources that are less capacitive, such as pass-

transistor switched tracks rather than tristate-buffered tracks. To investigate to what extent this is happening we used the same technique described in Section 6.3 to examine the relationship between wire capacitance and switching activity of nets routed with the power-aware router. Again we saw that nets with high switching activity are more likely to have a low capacitance when the power-aware routing algorithm is used, compared to when the baseline routing algorithm is used.

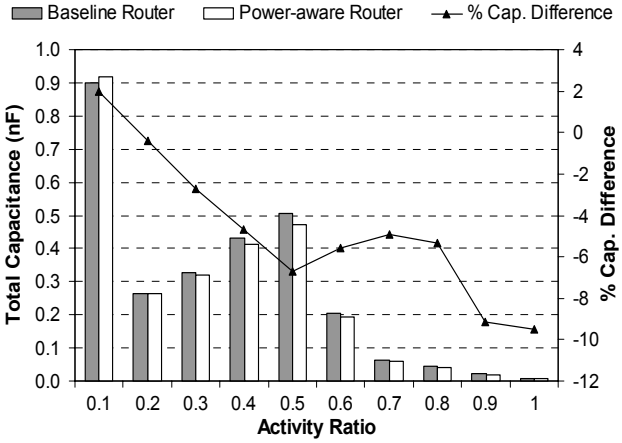

Figure 5. P-VPR Router (wire cap. vs. switching activity).

The power-aware router reduces global routing power by 6.2% compared to the baseline router. The critical-path delay, however, increases by 3.8%, again counteracting much of the power savings.

8. COMBINED RESULTS

The four previous sections considered each FPGA CAD algorithm in isolation in order to determine how suitable each algorithm is to energy minimization. This chapter, however, combines the power-aware algorithms described in the previous sections in order to examine the interactions between the savings of each power-aware algorithm. The results obtained for all sixteen possible CAD algorithm combinations are summarized in Table 4.

The energy reduction obtained when all the power-aware algorithms are combined is 22.6%. If the reductions of each stage were perfectly cumulative, the total reduction would be 25.8% (sum of the individual reductions). In other words, the reductions of the entire power-aware CAD flow are mostly cumulative, with only 3.2% overlap. To further investigate this, we examine the overlap between each power-aware algorithm separately.

For example, consider the interaction between the power-aware technology mapping and clustering algorithms. By itself, the power-aware technology mapping algorithm leads to a 7.6% reduction in energy. The power-aware clustering algorithm, by itself, leads to a 12.6% reduction in energy. Experimentally, by combining the two enhanced algorithms, we obtained an improvement of 17.6% (compared to 20.2% if the reductions had been perfectly cumulative). In other words, there is an overlap of 2.6% between the reductions achieved by the technology-mapper and the reductions achieved by the clusterer.

Using the same approach, the overlap for all remaining power-aware pairings was determined. The results are shown in Table 3.

Table 3. Overlap between power-aware algorithms.

Overlap (%)	Emap	P-T-Vpack	P-VPR Placer
P-VPR Router	0.27	0.04	-0.10
P-VPR Placer	0.39	-0.33	
P-T-Vpack	2.61		

The results suggest that most of the overall overlap occurs between the technology mapping and clustering algorithms. The overlap between the other algorithms is very small. A negative overlap implies that combining the algorithms introduces additional energy reductions; however, the negative values in Table 3 are very small and can be attributed to variance in the experimental results. It is intuitive that most of the overlap occurs between the technology mapping and clustering algorithms since the two algorithms account for most of the overall energy reduction. The overlap occurs when the technology mapping algorithm reduces the size of the netlist, leaving fewer wires for the clustering algorithm to work with. Generally, the overlap between the two stages increases proportionally with respect to the reductions of the technology mapper. This trend is illustrated in Figure 2, where each point corresponds to one benchmark circuit and the line is a linear regression trend line.

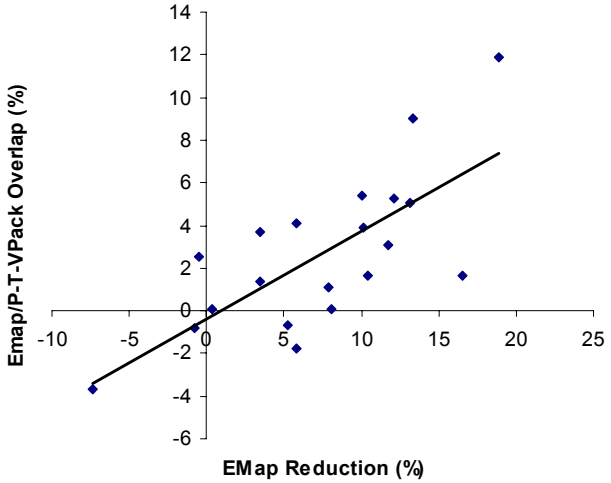

Figure 2. EMap/P-T-VPack Overlap.

Although not shown, the interactions between all the algorithms are similar; however, the effect is less dramatic since the reductions of the placement and routing algorithm are less significant.

9. CONCLUSION

In this paper, we have investigated the interactions between various stages in the FPGA CAD flow. We considered technology-mapping, clustering, placement, and routing. We have found that (1) the technology mapping and clustering algorithms were the most effective at reducing power, (2) the overlap between the energy savings achieved during each of the CAD stages is small. Of course, the numerical results are specific for our algorithms; however, we expect that other power-aware algorithms would produce similar conclusions. We have not yet considered high-level synthesis, but we expect that the energy savings achieved there could be significant.

Table 4. Combined Results (Energy (nJ)).

Mapper	Baseline	Baseline	Baseline	Baseline	Baseline	Baseline	Baseline	Baseline	Power	Power	Power	Power	Power	Power	Power	Power
Clusterer	Baseline	Baseline	Baseline	Baseline	Power	Power	Power	Power	Baseline	Baseline	Baseline	Baseline	Power	Power	Power	Power
Placer	Baseline	Baseline	Power	Power	Baseline	Baseline	Power	Power	Baseline	Baseline	Power	Power	Baseline	Baseline	Power	Power
Router	Baseline	Power	Baseline	Power	Baseline	Power	Baseline	Power	Baseline	Power	Baseline	Power	Baseline	Power	Baseline	Power
alu4	1.64	1.58	1.62	1.56	1.56	1.51	1.52	1.48	1.58	1.54	1.56	1.52	1.53	1.50	1.52	1.47
apex2	1.70	1.64	1.64	1.59	1.50	1.44	1.45	1.40	1.60	1.53	1.54	1.48	1.37	1.31	1.32	1.26
apex4	0.94	0.93	0.90	0.88	0.83	0.81	0.79	0.78	0.95	0.94	0.90	0.89	0.86	0.84	0.81	0.80
bigkey	2.34	2.29	2.33	2.29	2.40	2.36	2.39	2.37	2.51	2.47	2.49	2.45	2.48	2.44	2.46	2.42
clma	8.56	8.15	8.15	7.77	7.39	7.03	6.90	6.60	7.56	7.25	7.39	7.00	6.66	6.39	6.40	6.09
des	3.12	3.05	3.08	3.02	3.13	3.03	3.08	2.98	3.15	3.07	3.12	3.02	3.13	3.03	3.04	2.96
diffeq	0.90	0.87	0.88	0.84	0.76	0.73	0.73	0.71	0.79	0.77	0.78	0.75	0.69	0.68	0.66	0.65
dsip	2.22	2.21	2.19	2.17	2.04	2.00	2.01	2.00	2.21	2.18	2.21	2.16	2.03	2.00	2.01	1.98
elliptic	2.73	2.63	2.61	2.52	2.07	2.02	1.93	1.89	2.21	2.14	2.10	2.04	1.88	1.83	1.76	1.70
ex1010	2.85	2.78	2.81	2.73	2.46	2.41	2.31	2.27	2.56	2.52	2.52	2.46	2.21	2.18	2.07	2.04
ex5p	1.01	0.98	0.98	0.96	0.87	0.86	0.85	0.82	0.97	0.95	0.94	0.91	0.87	0.87	0.85	0.84
frisc	1.91	1.86	1.76	1.72	1.46	1.42	1.36	1.33	1.66	1.61	1.52	1.49	1.38	1.36	1.24	1.22
misex3	1.38	1.37	1.37	1.32	1.24	1.20	1.18	1.14	1.31	1.26	1.29	1.25	1.16	1.13	1.13	1.08
pdc	2.69	2.62	2.54	2.50	2.32	2.29	2.25	2.19	2.47	2.42	2.32	2.31	2.11	2.08	2.02	1.98
s298	2.23	2.15	2.17	2.11	1.76	1.72	1.70	1.68	2.10	2.04	2.06	2.01	1.72	1.69	1.70	1.66
s38417	7.43	7.27	7.25	7.08	6.60	6.46	6.48	6.37	6.69	6.55	6.59	6.44	6.26	6.13	6.13	5.99
s38584.1	5.89	5.76	5.82	5.68	5.19	5.06	5.09	4.97	4.92	4.83	4.87	4.78	4.31	4.22	4.24	4.13
seq	1.64	1.58	1.59	1.51	1.40	1.36	1.34	1.30	1.47	1.42	1.42	1.35	1.30	1.24	1.24	1.19
spla	2.02	1.96	1.91	1.84	1.73	1.69	1.59	1.57	1.86	1.83	1.77	1.72	1.59	1.56	1.51	1.49
tseng	0.96	0.94	0.94	0.92	0.85	0.83	0.83	0.81	0.84	0.81	0.83	0.81	0.77	0.75	0.75	0.73
Geo. Mean	2.18	2.12	2.11	2.05	1.90	1.85	1.83	1.79	2.01	1.96	1.95	1.90	1.79	1.75	1.73	1.68
% Diff.	0.00	-2.63	-2.99	-5.72	-12.6	-14.8	-15.9	-17.9	-7.59	-9.95	-10.19	-12.6	-17.6	-19.5	-20.6	-22.6

10. REFERENCES

[1] Anderson, J., and Najm, F.N., Power-Aware Technology Mapping for LUT-Based FPGAs, IEEE Intl. Conf. on Field-Programmable Technology, pp. 211-218, December 2002.

[2] Betz, V., and Rose, J., Architecture and CAD for the Speed and Area Optimization of FPGAs, *Ph.D. Dissertation* University of Toronto, 1998.

[3] Chen, K.C., Cong, J., Ding, Y., Kahng, A.B., and Trajmar, P., DAG-Map: Graph-Based FPGA Technology Mapping for Delay Optimization, IEEE Design and Test of Computers, September 1992, pp.7-20.

[4] Cong., J., and Ding, Y., FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs. IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 13(1):1-12, 1994.

[5] Cong, J., and Hwang, Y. Simultaneous Depth and Area Minimization in LUT-Based FPGA Mapping, Proc. Int'l Symp. on FPGAs, Monterey, CA, pp. 68-74, February 1995.

[6] Cong, J., Wu, C., and Ding, E. Cut Ranking and Pruning: Enabling A General And Efficient FPGA Mapping Solution, Proc. Intl. Symp. on FPGAs, Monterey, CA, pp. 29-35, 1999

[7] Li, H., Mak, W-K., and Katkoori, S., LUT-Based FPGA Technology Mapping for Power Minimization with Optimal Depth, IEEE Computer Society Workshop on VLSI, Orlando, 2001, pp.123-128.

[8] Marquardt, A., Betz, V., and Rose, J., Using cluster-based logic blocks and timing-driven packing to improve FPGA speed and density, Proc. Intl. Symp. on FPGAs, Monterey, CA, pp. 37-46, February 1999.

[9] Marquardt, A., Betz, V., and Rose, J., Timing-Driven Placement for FPGAs, Proc. Intl. Symp. on FPGAs, Monterey, CA, pp. 203-213, February 2000.

[10] McMurchie, L., and Ebeling, C., PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs, Proc. Intl. Symp. on FPGAs, Monterey, CA, pp. 111-117, 1995.

[11] Najm, F., Low-pass Filter for Computing the Transition Density in Digital Circuits, vol. 13, no.9, pp. 1123-1131, September 1994.

[12] Poon, K., Yan, A., and Wilton, S., A Flexible Power Model for FPGAs, Intl. Conf. on Field-Programmable Logic and Applications, September 2002.

[13] Sentovich, E.M., et al., SIS: A System for Sequential Circuit Synthesis, UC Berkeley, Memorandum No. UCB/ERL M92/41, Electronics Research Laboratory, May 1992.

[14] Singh, A., and Malgorzata, M., Efficient Circuit Clustering for Area and Power Reduction in FPGAs, Proc. ACM/SIGDA Intl. Symp. on FPGAs, Monterey, CA, pp. 59-66, February 2002.

[15] K. Roy, "Power-Dissipation Driven FPGA Place and Route Under Timing Constraints", IEEE Transactions on Circuits and Systems, vol. 46, no. 5, pp. 634-637, May 1999.