ON THE INTERACTION BETWEEN POWER-AWARE COMPUTER-AIDED DESIGN ALGORITHMS FOR FIELD-PROGRAMMABLE GATE ARRAYS

by

Julien Lamoureux
B.Sc.C.E., University of Alberta, 2001

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

The Faculty of Graduate Studies
Department of Electrical and Computer Engineering

We accept this thesis as conforming to the required standard:

The University of British Columbia
June 2003

Department of Electrical & Computer Engineering

The University of British Columbia
Vancouver, Canada

Date July 3rd, 2003

ABSTRACT

ON THE INTERACTION BETWEEN POWER-AWARE COMPUTER-AIDED DESIGN ALGORITHMS FOR FIELD-PROGRAMMABLE GATE ARRAYS

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 gains (in terms of energy reduction) are to be made and whether these gains are cumulative. In this thesis, 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 gains obtained by our power-aware technology mapping, clustering, placement, and routing algorithms and investigate how each gain behaves when the algorithms are applied concurrently. The individual energy reductions 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 energy reduction was achieved during the technology mapping and clustering stages of the power-aware FPGA CAD flow. In addition, the gains were mostly cumulative when the individual power-aware CAD algorithms were applied concurrently with overall energy reductions of 22.6%.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First of all, I wish to thank my academic advisor, Dr. Steve Wilton, for the guidance, technical advice, and moral support that he provided throughout my Masters. From Dr. Wilton, I learned a great deal about how to conduct and present research, and about the ins and outs of a career in research.

I would also like to thank the other members of Dr. Wilton's research group, namely: Andy Yan, Danna Cao, Ernie Lin, James Wu, Kara Poon, Martin Ma, Noha Kafafi, and Steve Oldridge, for their helpful insights and for creating a friendly research environment. Special thanks are due to Kara Poon for her detailed FPGA power model and to Dr. Guy Lemieux for his helpful and thought-evoking comments.

I greatly appreciate the financial support that was provided by the Altera Corporation, the BC Advanced Systems Institute (ASI), Micronet R & D, and the Natural Sciences and Engineering Research Council of Canada (NSERC). Without their support, this work would not be possible.

Finally, I would especially like to thank my family for their support and encouragement throughout my years of schooling, and Sasha Ransom for being a continual source of motivation and confidence.

Chapter 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 chips 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 wires and programmable switches. These wires are laden with parasitic capacitance. During high-speed operation, the switching of these wires causes significant power dissipation. Moreover, the programmable logic blocks used within FPGAs to implement user circuit functionality consume significantly more power than the fixed logic used within Application Specific Integrated Circuits (ASICs). Already, many FPGA vendors report that power dissipation is one of the primary concerns of their customers [56].

FPGA power consumption can be reduced by optimizing the circuitry and the architecture of the FPGA's programmable fabric, or the Computer-Aided Design (CAD) tools used to map circuits onto the FPGA. This thesis focuses on optimizing the FPGA CAD flow for power.

1.1 Research Goals

The FPGA CAD flow is comprised of a sequence of algorithms, namely: technology mapping, clustering, placement, and routing. These algorithms perform a series of transformations in order to map user circuits onto FPGAs. Each algorithm can be enhanced to minimize the power consumption of the final circuit implementation. There have been several low-power FPGA CAD algorithms described in previous works [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]. However, these have all been "point solutions", in that each considered only a single CAD algorithm. This thesis examines the entire FPGA CAD flow (from technology to routing) in order to understand the interaction between power-aware FPGA CAD algorithms. Specifically, the purpose of this thesis is to answer the following two questions:

1. What stages of the FPGA CAD flow are most suited to power minimization? This thesis focuses on technology mapping, clustering, placement and routing. Although, high-level synthesis algorithms would also be amenable to power minimization, it has not yet been investigated.

2. Are the power savings from individual power-aware stages cumulative? In other words, do the gains at one stage impact the gains that can be achieved in subsequent stages?

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

1.2 Research Approach

To answer the above questions, we enhance each algorithm in the FPGA CAD flow to minimize the power dissipated by circuits that are mapped onto FPGAs. We then investigate the individual and combined gains of the algorithms using an experimental framework that is the same for each algorithm. The framework begins with a baseline FPGA CAD flow consisting of well-established algorithms, as shown in Figure 1.1. The baseline CAD flow consists of CutMap [11], T-VPACK [12], and VPR [13, 14, 15]. These algorithms are representative of algorithms used in commercial FPGA CAD flows.

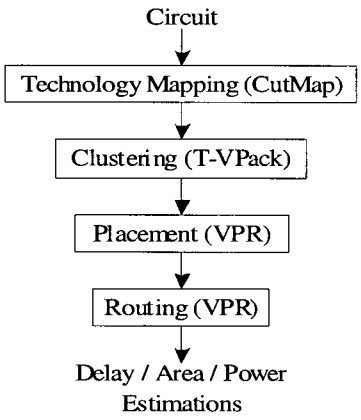


Figure 1.1: The baseline FPGA CAD flow.

To investigate the influence of each CAD stage on power 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 the impact of each stage on the power reduction. Then, we replace two or more of the baseline CAD stages with their power-aware counterparts to investigate the overlap between the gains of each stage. 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.

The gains of the power-aware algorithms are determined empirically by mapping a set of benchmark circuits with the given CAD flows and comparing the results to those obtained with the baseline CAD flow. Detailed models [14, 16] calculate the power and delay of the benchmark circuits when they are mapped onto FPGAs. Regardless of which power-aware algorithm is being evaluated, speed and power calculations are made after the routing stage of the CAD flow. This provides for much more accurate estimates than would be possible during earlier stages of the CAD flow, since only after routing can we accurately estimate the resistance and capacitance associated with each net.

1.3 Thesis Organization

This thesis is organized as follows. Chapter 2 gives and overview of FPGAs and summarizes the low-power FPGA CAD algorithm techniques employed in previous works. Chapter 3 describes the experimental framework we use to evaluate the performance of the power-aware FPGA CAD algorithms. Chapters 4 to 7 describe new power-aware technology mapping, clustering, placing, and routing algorithms, and the results that were obtained. Then, Chapter 8 combines the power-aware algorithms and examines the interaction between their gains. Finally, Chapter 9 summarizes the conclusions drawn throughout this thesis and provides suggestions for future work.

Chapter 2

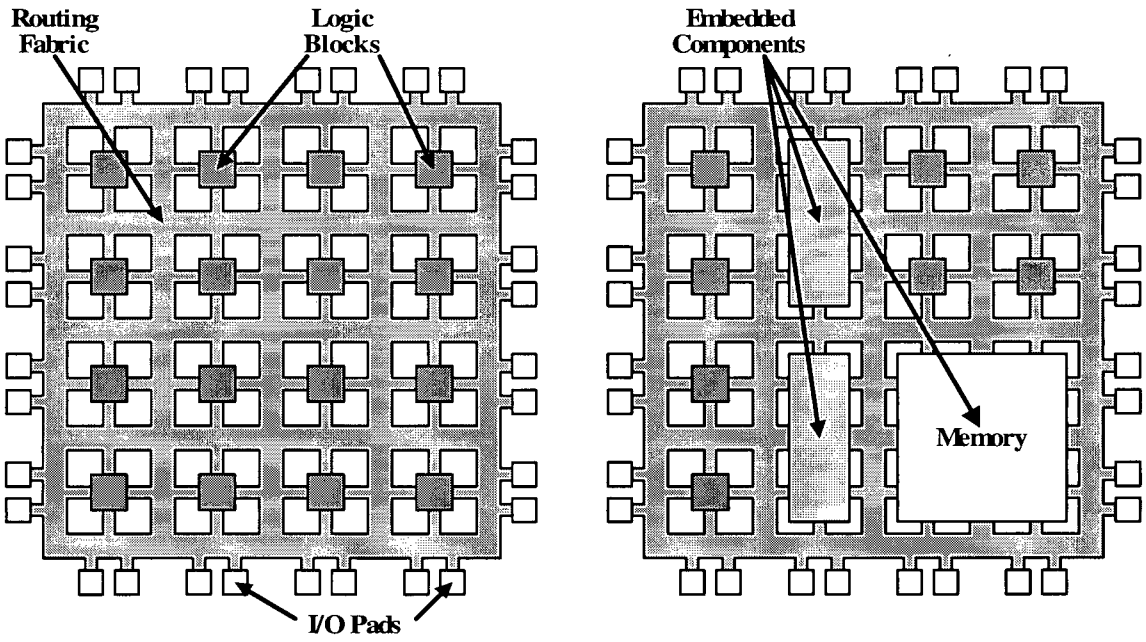
BACKGROUND AND PREVIOUS WORK

This chapter begins with an overview of Field-Programmable Gate Arrays (FPGAs). Specifically, it describes the architecture of the FPGA's programmable fabric and the CAD algorithms used to map user circuits onto an FPGA. It then summarizes the power-aware CAD algorithm techniques employed in previous works to reduce the power consumed by circuits that are implemented on FPGAs. Finally, the focus and contributions of this thesis are presented in the context of this previous work.

2.1 FPGA Architecture

Field-Programmable Gate Arrays are Integrated Circuits (ICs) whose functionality is programmed after fabrication. They consist of configurable logic blocks and I/O blocks that are interconnected by a configurable routing fabric. By configuring the logic blocks and routing fabric correctly, any digital *user circuit* can be implemented. In addition to these fundamental components, modern FPGAs also have embedded memories, embedded arithmetic logic units, and embedded processors, as shown in Figure 2.1 (b). Although this research focuses only on the programmable core of the FPGA, shown in Figure 2.1 (a), the results obtained are applicable to an FPGA with embedded components as well.

FPGAs are configured to implement user circuits by writing to the configuration memory that is embedded within the FPGA. Configuration memory is spread throughout the FPGA and it defines the logical function of each configurable logic block and the connections within the configurable routing fabric. Although other methods exist, FPGA configuration memory is typically implemented using static RAM (SRAM). Other technologies used to implement configuration memory include antifuses [17] and floating gate transistors [18]. However, this thesis focuses on SRAM-based FPGA devices exclusively, since SRAM-based FPGAs are the most common in commercial FPGAs.



a) A generic FPGA [14].

b) A modern FPGA with embedded components.

Figure 2.1: Conceptual FPGA models [16].

The FPGAs illustrated in Figure 2.1 are called *island-style* FPGAs, since the logic blocks resemble islands in a sea of configurable routing. The logic blocks are typically arranged in a grid and are surrounded by horizontal and vertical routing channels. Island-style FPGAs are the

most common in commercial FPGAs. The logic and routing resources of island-style FPGAs are described in the following two subsections.

2.1.1 Configurable Logic Blocks

Logic blocks implement the logical component of a user circuit. Since FPGAs must be flexible enough to implement any user circuit, FPGA logic blocks must be capable of implementing a wide range of logical functions. To achieve this flexibility, most commercial FPGAs use lookup-table (LUT) based logic blocks. A LUT with K inputs (K-LUT) contains 2^K configuration bits and can implement any K-input function (or gate). Using LUTs with many inputs (large K) reduces the number of LUTs required to implement a user circuit and subsequently reduces routing demands; however, it increases the area of the K-LUTs exponentially. By examining these speed, area, and routability tradeoffs, previous works have shown that 4-input LUTs result in the fastest and densest FPGAs [19, 20].

Early FPGAs had logic blocks that consisted of a LUT, a flip-flop, and local interconnect. This simple structure, called a *logic element* (LE), is illustrated in Figure 2.2. To enhance the functionality of the logic blocks, multiple LEs are combined into each logic block with additional local interconnect. This larger structure, called a *cluster*, is also illustrated in Figure 2.2. The advantages of clusters are similar to those of large LUTs: fewer logic blocks, less global routing, and better performance. However, the area penalty incurred by a cluster is much smaller than that of a large LUT. Modern FPGAs typically contain between 4 and 10 logic elements per cluster.

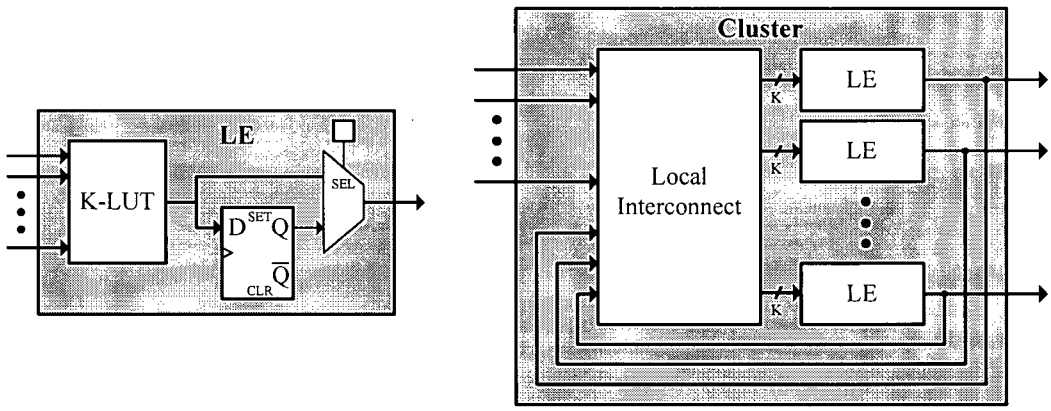


Figure 2.2: A generic FPGA logic block [21].

2.1.2 Configurable Routing Fabric

The FPGA routing fabric connects internal FPGA components such as logic blocks and I/O blocks. The performance of an FPGA is largely determined by the FPGA's routing architecture since routing accounts for the majority of the area, delay, and power of the FPGA. The island-style FPGA routing fabric consists of pre-fabricated wiring segments and programmable switches. The set of switches used to connect a logic block to an adjacent routing channel is called a *connection block*. Similarly, the set of switches used to connect intersecting routing channels is called a *switch block*. Figure 2.3 illustrates these various routing structures.

The structure of these individual routing components can be parameterized by *segment distribution*, *connection block topology*, and *switch block topology*, respectively. Segment distribution describes the lengths of the wire segments in the routing channels. Longer wire segments span multiple blocks and require fewer switches, thereby reducing routing area and delay. However, they also decrease routing flexibility, which reduces the probability that a user circuit can be routed successfully. Modern FPGAs commonly use a combination of long and short wires in order to balance this tradeoff. Connection and switch block topology describes the

interconnection pattern within these blocks. In terms of routability, fully populated blocks (that is, blocks in which any incident pin can be connected to any other incident pin) would be optimal. However, in terms of area, the cost would be prohibitive. Previous work [19, 22, 23, 67] has shown that connection and switch blocks still provide good routability even when only sparsely populated.

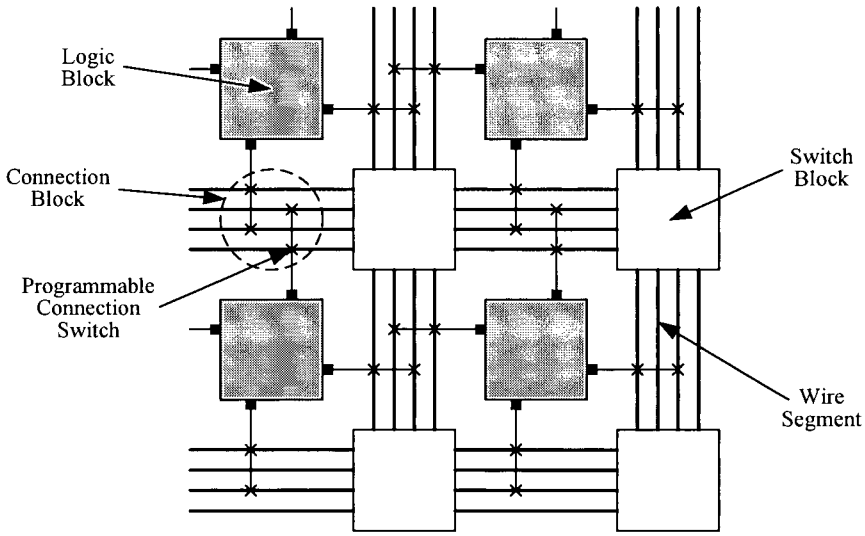


Figure 2.3: An island-style FPGA [14].

The programmable SRAM-based switches within the connection blocks and the switch blocks can be implemented using either pass-transistors or tri-state buffers, as illustrated in Figure 2.4. Pass-transistor switches require less area and dissipate less power than tri-state buffer switches. However, tri-state buffer switches are faster for connections that span many segments. Routing architectures commonly use a combination of tri-state buffer and pass-transistor switches to provide more selection for routing signals.

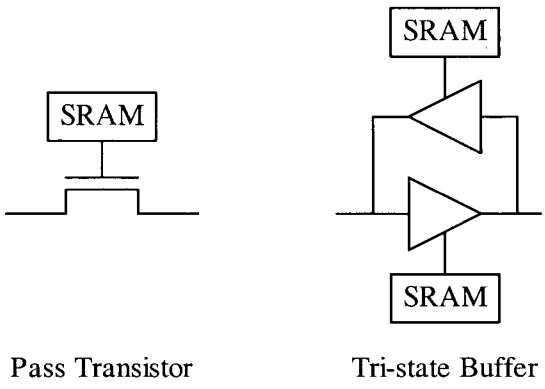


Figure 2.4: Two types of programmable switches used in SRAM-based FPGAs [14].

Global networks, such as clock and reset networks, are implemented with dedicated routing tracks which are separate from the configurable routing. Like other integrated circuits, FPGA clock distribution networks are designed to minimize skew in order to maximize system performance. Current FPGAs feature programmable phase-locked loops (PLLs) and delay-locked loops (DLLs) which allow users to multiply, divide, or shift external clock signals to synchronize with internal clock signals. They also support multiple clock signals, allowing users to manage several on and off-chip clock domains. Clock multiplexers and clock management circuitry are also available for disabling parts of the clock network to reduce power consumption. In order to achieve the best results, power-aware FPGA CAD tools must consider these features; however, the modeling and application of these networks is beyond the scope of this thesis. For simplicity, this thesis assumes a global H-tree clock distribution network.

2.2 FPGA CAD Flow

Since large FPGAs contain millions of configuration memory bits, the state of each bit (0 or 1) must be determined using Computer-Aided Design (CAD) tools. These CAD tools transform high-level circuit descriptions into configuration bitstreams. Circuits are mapped onto the FPGA by writing this bitstream into the configuration memory of the FPGA, which defines the state of

the FPGA's routing switches and the functionality of the FPGA's logic blocks. The CAD flow is divided into a sequence of stages, illustrated in Figure 2.5.

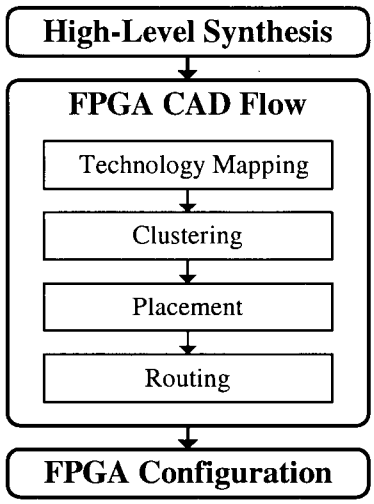
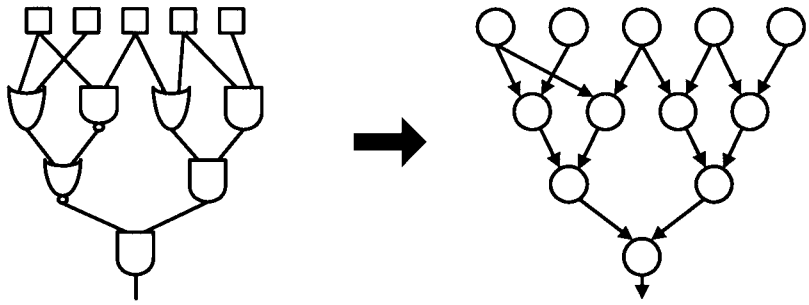


Figure 2.5: The FPGA CAD Flow.

High-level synthesis is the first stage of the CAD flow. It transforms a high-level circuit description into a Boolean network consisting of basic logic gates and flip-flops. After high-level synthesis, the *FPGA CAD flow* maps this Boolean network onto the configurable fabric of the FPGA. The FPGA CAD flow has four stages: technology mapping, clustering, placement, and routing. These stages are described in the following subsections.

2.2.1 Terminology

Before describing each stage of the FPGA CAD flow, we review some terminology defined in [11,24] to describe Boolean networks (user circuits). 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 as shown in Figure 2.6.



A Boolean network

An equivalent directed acyclic graph (DAG)

Figure 2.6: A directed acyclic graph representation of a Boolean network.

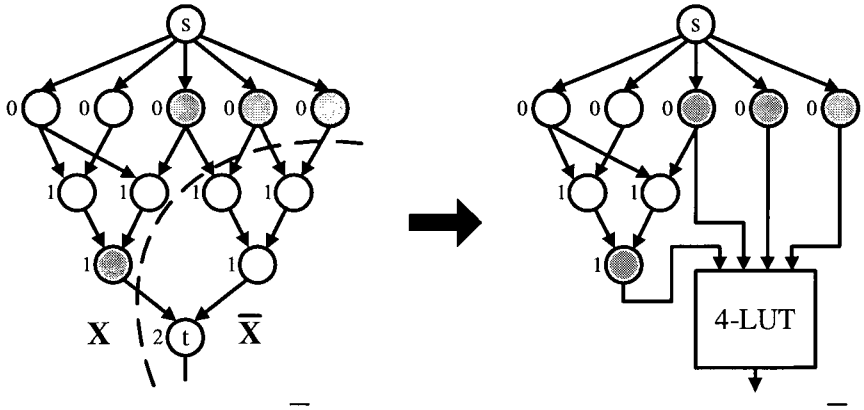
Given a network $N=(V(N), E(N))$ with a source s and a sink t, a cut $(X, \overline{X})$ is a partition of the nodes in $V(N)$ such that $s \in X$ and $t \in \overline{X}$ (see Figure 2.7). The *cut-size* is the number nodes in X that are adjacent to the nodes in $\overline{X}$. A cut is *K-feasible* 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 is denoted $output(v)$. 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 all 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. A *cone* rooted at node v, denoted C_v, is a subgraph consisting of v and its predecessors nodes. A fanout-free cone (FFC) at v, denoted FFC_v, is a cone of v that does not contain a node that fans-out to a node external to the cone. Finally, a maximum fanout-free cone (MFFC) at v, denoted $MFFC_v$, is the fanout-free cone at v with largest number of nodes.

Additional terminology, used to describe the timing information of Boolean networks, include *required time*, *arrival time*, *slack*, and *criticality*. The required time for a node v, denoted

required(v), is the amount of time that a signal has to make a transition before becoming illegal. Similarly the arrival time for a node v, denoted *arrival*(v), is the amount of time that a signal takes to make a transition. The slack of a node v, denoted *slack*(v), is the difference between the required time and the arrival time of the node. The nodes along the critical paths of a Boolean network have a slack of zero since their required time and arrival time are equal. The remaining nodes have a slack that is positive. Finally, the criticality of a node v, denoted *criticality*(v), is a measure of how close a node is to being on the critical path.

2.2.2 Technology Mapping

Technology Mapping is the first stage of the FPGA CAD flow that we consider. This stage transforms the Boolean network that was generated during high-level synthesis into a LUT and flip-flop network, thereby allowing it to be implemented on LUT-based FPGAs. K-bounded Boolean networks can be transformed into LUT networks since K-input lookup tables are capable of implementing any logical function with K-inputs or less. Since a K-LUT can implement any K-input function, the task of mapping a Boolean network to a LUT network is equivalent to choosing a set of K-feasible cuts that include all the nodes in the network. Figure 2.7 illustrates a cut (X, $\overline{X}$) of a Boolean network and the corresponding LUT mapping.



A 4-feasible cut (X, $\overline{X}$) LUT mapping of cut (X, $\overline{X}$)

Figure 2.7: The lookup table equivalent of a cut.

Technology mapping algorithms typically aim to minimize the delay, area, or power requirements of the final circuit. Circuit delay can be minimized during technology mapping by minimizing the depth of the LUT network. The depth of a LUT network is the length of the longest path from any input to any output of the network. Minimizing the length of the longest path minimizes the critical-path delay of a circuit. The FlowMap algorithm, described in [67], was a breakthrough in the area of LUT-based FPGA technology mapping. Until FlowMap, a number of heuristic algorithms were proposed, but none guaranteed optimal depth solutions for general Boolean networks. It was proven in [67] that an optimal depth solution can be found for any circuit in polynomial time. The key step in the algorithm incorporates the *Max-Flow Min-Cut theorem* [24] to compute a minimum height K-feasible cut for each node in a network. Compared to previous technology mapping algorithms, FlowMap reduced the depth and the area of circuits by 7% and 50% respectively.

After FlowMap, various depth-optimal algorithms were proposed to further minimize area [11, 25, 26]. The CutMap algorithm, which is described in [11], produces area efficient depth-optimal circuits by minimizing node duplication. Node duplication occurs when a node is encapsulated within more than one LUT. This is shown with an example in Figure 2.8. Although node duplication is necessary for depth optimization, excessive node duplication increases circuit area. CutMap avoids node duplication by favoring cuts in which the nodes that are inputs to the cut have either previously been cut, or are likely to be cut. Nodes with large MFFC are likely to be cut since encapsulating their predecessor nodes does not cause node duplication. CutMap produces depth-optimal circuits that require, on average, 20% fewer LUTs

than FlowMap. This reduction in area also helps to reduce power consumption since the circuit requires fewer FPGA resources.

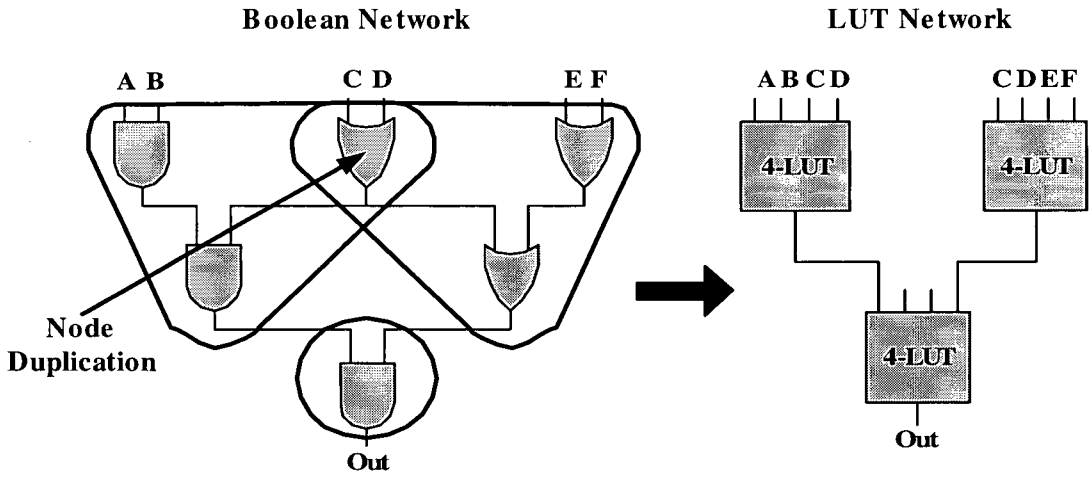


Figure 2.8: LUT-based technology mapping.

Several previous works have focused on minimizing power consumption directly [1, 2, 3, 4, 5, 6]. These power-aware technology mapping algorithms typically minimize power by hiding wires with high switching activity within LUTs. This reduces dynamic power dissipation since the internal wire capacitance of LUTs is significantly smaller than the external wire capacitance of the routing fabric that connects them. The power-aware algorithm described in [4] is very similar to FlowMap. Like FlowMap, the algorithm finds minimum height K-feasible cuts for critical nodes in order to optimize circuit depth. However, for non-critical nodes, the algorithm finds minimum weight K-feasible cuts in order to minimize power consumption. They define a minimum weight K-feasible cut $(X_v, \overline{X}_v)$ for node v as a K-feasible cut such that the maximum estimated power consumption of the nodes X_v that provide inputs to the nodes in $\overline{X}_v$ is minimized. The algorithm estimates the power consumption of these nodes recursively using the following expression:

$$ep(v) = K_p \cdot (C_{in} + C_{out}) \cdot P(v) + \sum_{u_i \in input(v)} ep(u_i),$$

where K_p is a constant, C_{in} and C_{out} are input and output LUT capacitance, and $P(v)$ is the switching probability of node v. This cost function favors implementations in which LUTs have low input switching activities and capture high activity nets internally. The algorithm also guarantees depth-optimal circuits since only non-critical nodes are affected; however, it is only effective for circuits with many non-critical nodes. Power savings for circuits with little slack tend to be small.

The power-aware technology mapping algorithm described in [5] uses a technique called *cut-enumeration*. This technique involves finding multiple mapping solutions (cuts) for each node in the network and then selecting the best one for each node using a cost function. Their implementation has three phases. The first stage calculates the switching activity of each node in the circuit using the transition density model [27], which is described in Chapter 3. The second stage then enumerates at most p cuts for each node in topological order (beginning from the primary inputs), where p is a user-specified positive integer. Enumerating only p cuts for each node reduces the complexity of the algorithm without significantly affecting the final solution. The third stage uses information from the first two stages to select the best cut for each node and to produce the final mapping solution. It uses the following cost function to determine the best mapping solutions:

$$P_{avg}(N) = \frac{1}{2} \cdot V_{dd}^2 \cdot \left[\sum_{p_i \in PI} |output(p_i)| \cdot C_{in} \cdot D(p_i) + \sum_{l_i \in LUT} \left(C_{out} + |output(l_i)| \cdot C_{in}\right) \cdot D(l_i) \right]$$

where V_{dd} is the supply voltage, $|output(x)|$ is the number of LUTs receiving input from the node x, C_{in} and C_{out} are the input and output capacitance of a LUT, and $D(v)$ is the estimated switching

activity of node v. Power savings of 14% were reported for this algorithm compared to previous methods. The algorithm, however, does not optimize circuit depth. Thus, the resulting circuits are likely to be slower than depth-optimal circuits.

The most recent work, described in [6], explores the tradeoff between circuit depth and power consumption. Specifically, it shows that node duplication, which is necessary for depth optimization, is costly in terms of power consumption. Previous empirical work has shown that switching activity in combinational circuits typically decreases quadratically with circuit depth [28]. Replicating a node reduces its fanout size but it increases the fanout size of its fanin nodes. Since fanin nodes have lower depth than fanout nodes, node duplication tends to increase power consumption. The algorithm begins by estimating the switching activity of each node in the circuit using the *transition probability model* [29]. It then enumerates the set of all K-feasible cuts for each node in the circuit. After cut-enumeration, the algorithm re-traverses the network in topological order to select the best cut for each node. The cost function used to select the cuts has three components:

$$Cost(X_z, \overline{X_z}) = \alpha \cdot DCost(X_z, \overline{X_z}) + \beta \cdot PCost(X_z, \overline{X_z}) + \gamma \cdot RCost(X_z, \overline{X_z}),$$

where α, β, and γ are constant coefficients that reflect the relative importance of each component. $DCost(X_z, \overline{X_z})$ is the depth cost component, which is defined as the depth of node z in the subgraph that corresponds to the LUT mapping solution of cut $(X_z, \overline{X_z})$. $PCost(X_z, \overline{X_z})$ is the power cost component, which has two terms:

$$PCost(X_z, \overline{X_z}) = \sum_{v \in input(\overline{X_z})} [f_v + PCost(BestCut(v))] - \sum_{w \in \overline{X_z}} [f_w \cdot |output(w) \cap \overline{X_z}|]$$

where f_x represents the switching activity of the net driven by node x. The first term is the summation of the activities of the nets that fan into to cut $(X_z, \overline{X_z})$, while the second term is the summation of the activities of the nets that are encapsulated within the LUT that corresponds to cut $(X_z, \overline{X_z})$. Finally, $RCost(X_z, \overline{X_z})$ is the replication cost component. This component sums the activities of the fanin nets that are replicated and substracts the activities of the fanout nets that are encapsulated for each node that is replicated in the subgraph corresponding to cut $(X_z, \overline{X_z})$. Intuitively, the cost function balances the delay cost of increasing depth with the power cost of node duplication.

2.2.3 Clustering

Clustering is the second stage of the FPGA CAD flow. It produces a netlist of logic blocks from a netlist of LUTs and flip-flops. Most FPGA logic blocks contain between 4 and 10 LUT/flip-flop pairs called logic elements (LE) [30, 31]. The clustering algorithm partitions the input netlist of LEs into clusters, which can then be mapped into the logic blocks of the FPGA, as illustrated in Figure 2.9. The aim of clustering is to minimize the number of logic blocks and the number of connections between the logic blocks. Timing-aware clustering tools are also possible.

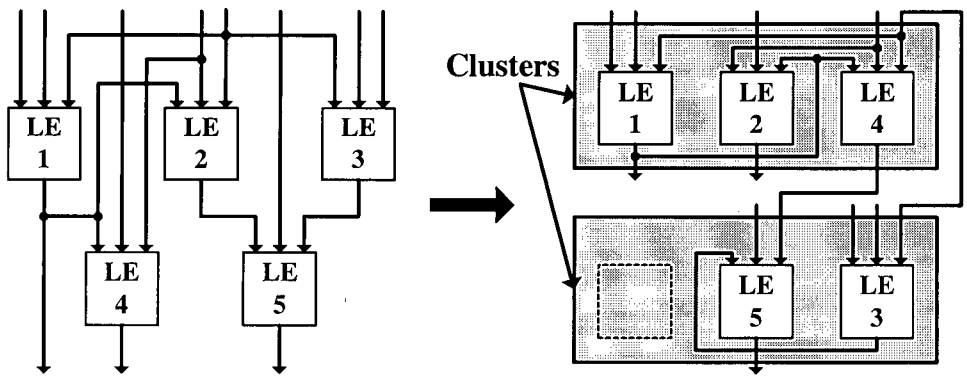


Figure 2.9: An example of clustering.

There are two general clustering approaches: bottom-up [12, 14, 32, 33, 34] and top-down [35, 36]. Bottom-up approaches build clusters individually, packing each cluster around a seed LE until it is full. Top-down approaches partition the LEs into clusters by successively subdividing the network or by iteratively moving LEs between partitions. Bottom-up approaches are typically faster and more simple than top-down approaches since they only consider local connectivity information and can easily satisfy logic block pin constraints. Top-down approaches offer the best solutions; however, their computational complexity can be prohibitive.

The VPack algorithm, described in [14], uses a bottom-up approach. LEs are packed one at a time. For each cluster, an attraction function is used to select a seed LE from the set of all LEs that have not already been packed. After packing a seed LE into the new cluster, a second attraction function selects new LEs to pack into the cluster. LEs are packed into the cluster until the cluster 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. The VPack algorithm is outlined in Figure 2.10.

```
UnclusteredBLEs = PatternMatchToBLEs(LUTs, Registers);
LogicClusters = NULL;

while(UnclusteredBLEs != NULL) {
        C = GetBLEwithMostUsedInputs(UnclusteredBLEs);

        while(|C| < N) {// cluster is not full
                BestBLE = MaxAttractionLegalBLE(C, UnclusteredBLEs);
                If (BestBLE == NULL) // no BLE can be added to this cluster
                        break;
                UnclusteredBLEs = UnclusteredBLE – BestBLE;
                C = C ∪ BestBLE;
        }
        if (|C| < N) { // cluster is not full – try hill climbing
                while(|C| < N) {
                        BestBLE = MinClusterInputIncreaseBLE(C, UnclusteredBLEs);
                        C = C ∪ BestBLE;
                        UnclusteredBLEs = UnclusteredBLEs – BestBLE;
                }
                if (ClusterIsIllegal(C))
                        RestoreToLastLegalState(C, UnclusteredBLEs);
        }
        LogicClusters = LogicClusters ∪ C;
}
```

Figure 2.10: Pseudo-code of the VPack algorithm [14].

The T-VPack algorithm [12, 15] is a timing-aware clustering algorithm that is based on VPack. The algorithm is identical to VPack, however, the attraction functions used to select the LEs to be packed into the clusters are different. The VPack seed function chooses LEs with the most used inputs, whereas the T-VPack seed function chooses LEs that are on the most critical path. VPack's second attraction function chooses LEs that share the most connections with the LEs already packed into the cluster. T-VPack's second attraction function has two components for an LE B being considered for cluster C:

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

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, α is a user-defined constant which determines the relative importance of the attraction

components, and G is a normalizing factor. The first component of T-VPack's second attraction function chooses critical-path LEs, and the second chooses LEs that share many connections with the LEs already packed into the cluster. By initializing and then packing clusters with critical-path LEs, the algorithm is able to absorb longer sequences of critical-path LEs into clusters. This minimizes circuit delay since the local interconnect within the cluster is significantly faster than the global interconnect of the FPGA. Experimentally, circuits clustered using T-VPack are 2.0% more energy efficient than circuits clustered using VPack. T-VPack is the baseline clustering algorithm in this thesis.

Although no previous work has focused on power during the clustering stage to the CAD flow, the algorithm recently described in [7] minimizes power indirectly by minimizing inter-cluster wiring. The algorithm minimizes inter-clustering wiring by (1) absorbing as many small nets into clusters as possible, and (2) depopulating clusters according to Rent's rule [37] in order to balance the amount of interconnect between clusters. Absorbing nets entirely into clusters reduces the overall net count, which simplifies routing. Small nets can more easily be absorbed entirely into clusters since they have fewer terminals. To balance the amount of interconnect between clusters, the algorithm limits the number of available pins using Rent's rule. The aim is to match the number of inputs of each cluster to the amount of logic implemented within the cluster in order to alleviate congestion.

Like the previous clustering algorithms, the above algorithm packs one cluster at a time using two cost functions. The seed function chooses LEs with the lowest *connectivity* value, which is defined as:

$$connectivity(LE) = \frac{separation(LE)}{degree(LE)^2}$$

where *separation* is the total number of pins of all the nets that are incident to *LE*, and degree is the number of nets incident to *LE*. LEs with a small *connectivity* value are more easily absorbed since the nets that are incident to it are not connected to many other LEs. The second attraction function is similar to VPack in that it chooses LEs that share nets with the LEs already in the cluster. However, to reward net absorption, the attraction is multiplied by a constant k, where k > 10, if adding the LE to the cluster fully absorbs a net. Compared with T-VPack, the algorithm reduces the routing area. Correspondingly, the power dissipated by global interconnect is also reduced.

2.2.4 Placement

The third stage of the FPGA CAD flow is placement. In this stage, physical locations are assigned to each logic block in the netlist produced by the clustering algorithm, as shown in Figure 2.11. Placement algorithms seek to simultaneously minimize routing demands and critical-path delays. Routing demands are reduced by placing highly interconnected logic blocks close together. Similarly, critical-path delays are minimized by placing logic blocks along critical-path nets close together.

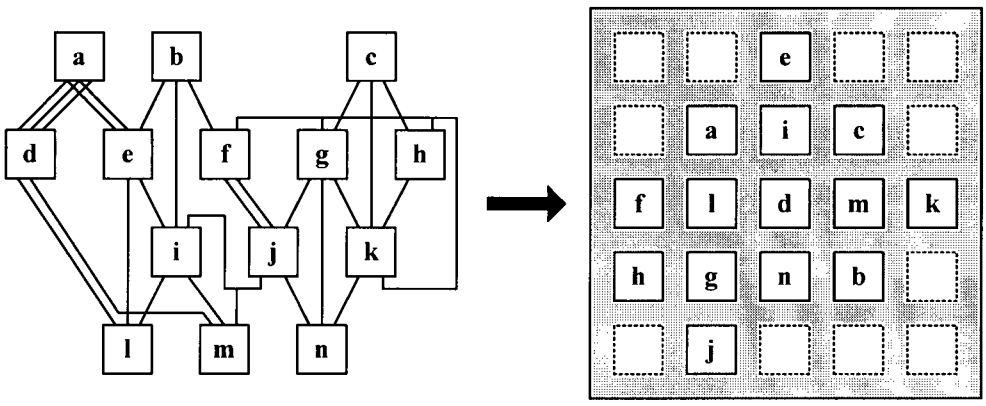


Figure 2.11: An example of placement.

There are three general placement approaches: min-cut [38, 39, 40], analytic [41, 42, 43], and simulated annealing [13, 14, 15, 44, 45, 46]. Although each technique has been shown to produce good results, simulated annealing-based placers are much more adaptable than other placers to new optimization goals and architectural changes. Pseudo-code describing a generic simulated annealing-based placement algorithm is shown in Figure 2.12.

```
S = RandomPlacement();
T = InitialTemperature();
R_limit = InitialR_limit();

while(ExitCriterion() == False) {                    /* Outer Loop */
        while(InnerLoopCriterion() == False) {       /* Inner Loop */
                S_new = GenerateViaMove(S, R_limit);
                ΔC = Cost(S_new) – Cost(S);

                r = random(0, 1);
                if (r < e^(-ΔC/T))
                        S = S_new; // accept sway
        }
        T = UpdateTemp(); // cool
        R_limit = UpdateR_limit();
}
```

Figure 2.12: Pseudo-code of a generic simulated annealing-based placer [14].

The algorithm starts with a random initial placement of the user circuit. Pairs of logic blocks are then randomly selected and then swapped repeatedly. 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.

T-VPlace [13], is a simulated annealing based placement algorithm which minimizes routing and critical-path delay. 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 [14]. 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 [14]. The total cost is the sum of the wiring cost and timing cost for all nets:

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

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

A power-aware placement and routing algorithm, described in [8], minimizes power by placing logic blocks connected by high-activity nets close together. Like T-VPlace, the algorithm is based on simulated annealing; however, the algorithm targets row-based FPGAs with antifuse

configuration memory instead of island-style FPGAs with SRAM configuration memory. The placement algorithm's cost function has four components:

$$C = W + a \cdot T + b \cdot P + c \cdot F$$

where, W is the total wire length, T is the critical-path delay, P is the overall power dissipation, and F is the extra cost associated to using an uncommitted feed through (a vertical connection). The a, b, and c weight factors determine the relative importance of each cost component. The first two components of the cost function are similar to those of the T-VPlace cost function. The wire length is estimated using bounding boxes, and delay is estimated using Elmore delay. The third component, however, estimates power by multiplying the wire length estimate of each net by the estimated activity of each net. The final component is specific to row-based FPGAs. Power reductions of up 40% were reported for the row-based placement and routing algorithm; however, the impact on critical-path delay was not considered.

2.2.5 Routing

The final stage of the FPGA CAD flow is routing. Routing determines how the connections between logic blocks are formed within the prefabricated configurable routing fabric. The aim of routing is to minimize critical-path delays and to avoid congestion. Congestion is avoided by routing connections as directly as possible and by balancing the usage of routing wires across the FPGA. Critical-path delays are minimized by giving priority to high-criticality nets when contention occurs between nets for a given routing resource.

There are two routing approaches: two-step routing [15, 47, 48, 49, 50, 51], which performs global routing and then detailed routing, and combined global-detailed routing [52, 53, 54, 55],

which performs global and detailed routing in a single step. Global routing determines which logic block pin and routing channel is used by each net, and detailed routing determines which wire segments within a routing channel are used by each net. Two-step routers are commonly used for ASICs; however, they are not typically used for FPGAs since the limited flexibility of the FPGA routing fabric makes detailed routing difficult after global routing constraints are applied. It is hard for the global FPGA routers to know if detailed routing is possible without actually performing the detailed routing.

The VPR router [15] uses a negotiated congestion-delay algorithm based on PathFinder [53]. 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 tracks are used by more than one net.

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

$$Cost(n) = Criticality(i, j) \cdot delay_{Elmore}(n) + (1 - Criticality(i, j)) \cdot b(n) \cdot h(n) \cdot p(n)$$

The cost function has a delay term and a congestion term. The delay term is the product of the normalized Elmore delay of node n and *Criticality(i,j)* as defined in Section 2.2.1. 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.

Like T-VPlace, the VPR router performs well in terms of delay and power, since minimizing timing reduces delay and minimizing congestion reduces the power dissipated by global routing. However, no previous work has focused specifically on minimizing power during the routing stage of the FPGA CAD flow.

2.3 Focus and Contribution of this Thesis

The goal of this research is to understand where the gains (in terms power reduction) can be expected within the FPGA CAD flow and how these gains interact. Understanding these issues is a key step towards creating power-aware FPGA CAD tools. All the power-aware enhancements described in Section 2.2 were developed and evaluated in isolation. In order to understand where power gains can be expected within the FPGA CAD flow, the algorithms must be evaluated using a common experimental methodology that is detailed enough to capture the improvements made by each stage. Similarly, in order to understand how the gains achieved in earlier stages affect the gains achieved in later stages, the power-aware algorithms must be applied in succession. Our approach involves enhancing a baseline FPGA CAD flow that is comprised of algorithms that are representative of those used in commercial FPGA CAD flows today. Using the detailed post-route power and delay models described in the following chapter, we determine the gains of the individual power-aware algorithms in Chapters 4, 5, 6, and 7 and of the combined algorithms in Chapter 8.

The contributions of this thesis are summarized as follows:

1. Developed new power-aware technology mapping, clustering, placement, and routing algorithms by enhancing existing algorithms using techniques employed in previous works or straightforward extensions of the baseline CAD algorithms.

2. Compared the gains of each power-aware algorithm using very detailed post-route power and delay models in order to determine which stages are most suited to power optimization.
3. Measured the gains when the power-aware algorithms are combined to determine if the individual gains are cumulative.

Chapter 3

EXPERIMENTAL METHODOLOGY

This chapter describes the experimental methodology employed to measure the individual and combined gains of the power-aware algorithms that are presented in this thesis. It begins by describing the requirements of the experimental methodology and why the experimental methodologies employed in previous works are inadequate for investigating the interaction between power-aware algorithms. It then presents the new experimental methodology that is employed in this thesis. Finally, it describes the detailed models that are used to estimate the power and delay of circuits after they are mapped onto an FPGA.

3.1 Motivation

The effectiveness of power-aware FPGA CAD algorithms can only be determined empirically because of the inherent complexity of the interactions between FPGA CAD algorithms, FPGA architectures, and the circuits that are mapped onto the FPGAs. Since FPGAs are intended to implement any circuit, the effectiveness of the algorithms must be averaged for a wide range of benchmark circuits that are reflective of circuits that are typically implemented on FPGAs. Ideally, power-aware FPGA CAD algorithms would be evaluated by mapping benchmark circuits onto real FPGAs and then measuring the power dissipated by the FPGAs during normal operation. However, this methodology is not possible since the CAD tools of commercial

FPGAs are proprietary. Instead, the process of mapping circuits onto FPGAs is typically modeled.

In previous works [1, 2, 3, 4, 5, 6, 7], the power models used to evaluate power-aware algorithms were overly simplified. In each of these works, power was modeled before placement and routing, when there is no information regarding the capacitance of the nets that are implemented within the routing fabric of the FPGA. The models either assume the same capacitance for each connection or estimates the capacitance of each net based on the fanout of the net. These models do not capture the gains that can be achieved during later stages of the FPGA CAD flow, when high activity nets are made shorter to reduce power.

Furthermore, the models employed in the previous works only consider the dynamic power dissipated within the global interconnect of the FPGA. Recent studies of FPGA power dissipation [16, 56] have reported that dynamic routing power accounts for only half of the power dissipated by FPGAs. The dynamic power dissipated within the logic blocks and the clock distribution network of the FPGA, as well as the overall static power dissipation, should also be considered. By not considering these sources of power dissipation, the gains reported in the previous works are likely exaggerated.

Finally, when comparing power-aware FPGA CAD algorithms, it is important to consider both power and critical-path delay. Power-aware algorithms that do not also consider timing are likely to produce circuits with longer critical-path delays than circuits produced using timing-aware algorithms. Using CAD algorithms to minimize power at the expense of delay is not

practical since equivalent results can be obtained by simply slowing the system clock. Therefore, the power-delay product (energy) metric is used to evaluate the performance of the power-aware algorithms that are described in this thesis.

In order to fairly compare the energy reductions at each stage of the FPGA CAD flow, the same experimental methodology must be employed at each stage. Specifically, the same benchmark circuits, models, and FPGA architectural assumptions should be used for each algorithm. The following section describes the experimental methodology used for each experiment in this thesis.

3.2 New Experimental Methodology

The experimental methodology used for this thesis employs detailed models to estimate the power and critical-path delay of the standard benchmark circuits after they are placed and routed onto an FPGA, as shown in Figure 3.1. The results are compared to the results from a baseline FPGA CAD flow, which consists of non power-aware algorithms that are representative of algorithms used in commercial FPGA CAD flows. Specifically, the baseline FPGA CAD flow consists of CutMap [11], T-VPACK [12], and VPR [13, 14, 15].

To investigate the influence of each algorithm on energy minimization, we replace baseline algorithms with power-aware algorithms. Initially, in Chapters 4 through 7, we replace only one baseline algorithm at a time, so that we can examine the impact of each individual algorithm on energy minimization. Then, in Chapter 8, we replace multiple baseline algorithms with their power-aware counterparts to investigate the interaction between the power-aware algorithms. In all cases, the power-aware algorithms we implement are representative of power-aware

algorithms that have either been published in the literature or are straightforward extensions of the baseline CAD algorithms.

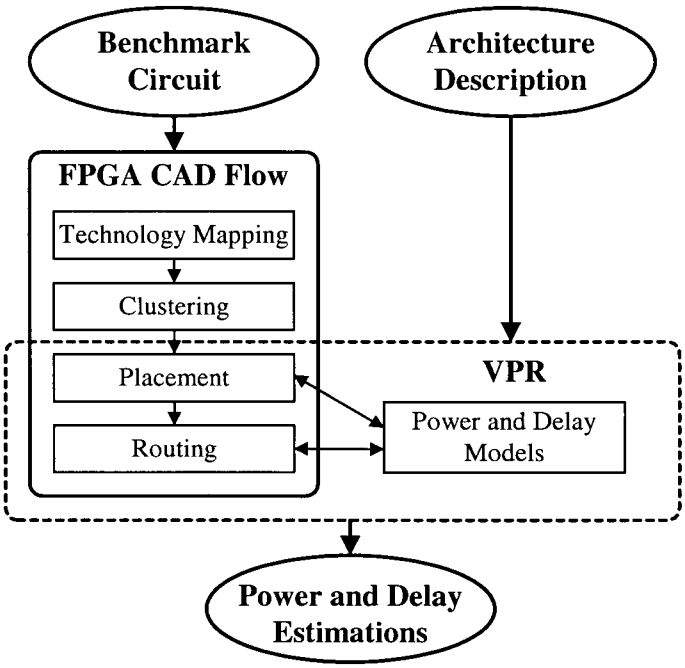


Figure 3.1: Experimental Framework.

The benchmark circuits are mapped onto FPGAs that are modeled within the VPR CAD tool. VPR builds a routing-resource graph that corresponds to the architecture under investigation. The routing-resource graph contains the connectivity and circuitry information that is needed by the placement and routing algorithms to optimize for power and delay. After routing, the power and delay models also access the routing-resource graph to obtain detailed information regarding the implementation of the circuit. This allows estimates to be much more accurate than estimates obtained during earlier stages of the CAD flow, since only after routing can the resistance and capacitance associated with each net be accurately modeled.

The benchmark circuits consist of 20 of the largest MCNC circuits. Namely, the MCNC circuits used are: alu4, apex2, apex4, bigkey, clma, des, diffeq, dsip, elliptic, ex1010, ex5p, frisc,

misex3, pdc, s298, s38417, s38584.1, seq, spla, and tseng. Half of the benchmark circuits are sequential circuits and the other half are purely combinational. The circuits range in size from 1858 to 14233 2-input gates. Each circuit was optimized in SIS using script.rugged [57] and then transformed into a network of 2-input gates using dmig [57].

The experimental FPGA architecture used in this thesis is chosen to be representative of commercial architectures. This architecture was shown to be efficient in terms of area and delay in [14]. There are 4 LEs per logic block and each LE has 4 inputs. The fraction of wires in each channel to which a logic block pin can connect to, denoted F_c, is 0.5, and the number of wires to which each incoming wire can connect to in a switch block, denoted F_s, is 3. All the wire segments have a length of four; half are buffered and the other half are unbuffered. The channels are uniform (same number of tracks in each channel) and unbiased (same number of tracks in the horizontal and vertical channels). Finally, the channels have at least 20% more routing tracks than the minimum routeable width, and are fixed when comparing algorithms. Using fixed channel widths for each given benchmark circuit produces unbiased results, since the architecture is the same for both results.

3.3 The Delay Model

To estimate critical-path delays, we use the delay model that was originally incorporated into the VPR CAD tool. The model combines Elmore delay and HSPICE characterization to produce detailed delay estimations of user circuits after routing. VPR estimates the critical-path delay of a circuit by calculating the delay of circuit elements along its slowest path. HSPICE is used to determine the intrinsic delay of logic blocks, connection blocks, and routing buffers. The delay of these elements can be pre-characterized since they are independent of placement and routing.

Elmore delay is used to determine the delay of the remaining routing circuitry. These calculations can only be performed after routing since the resistance and capacitance of the routing resources used by each path is required. The Elmore delay of a path formed between logic or I/O blocks is [58]:

$$\sum_{i \in path} R_i \cdot C(subtree_i) + T_{d,i}$$

where i is the ith routing element of the path, R_i is its resistance, $C(subtree_i)$ is its subtree capacitance, and $T_{d,i}$ is the intrinsic delay of a buffer if element i is a buffer, and 0 otherwise. Figure 3.2 illustrates the RC equivalent circuits assumed by VPR for wires, pass-transistor switches, and tri-state buffer switches.

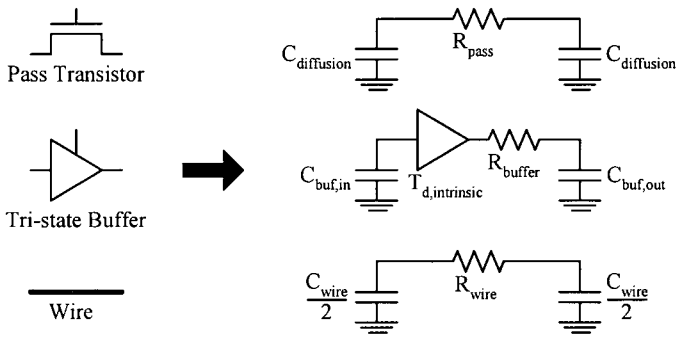


Figure 3.2: Equivalent circuit for FPGA routing elements [14].

The Elmore delay of an RC-tree can be computed in linear time and it has been shown to have good fidelity [14]. The HSPICE characterizations and the resistance and capacitance values used by VPR are based on the TSMC 0.18 μm, 1.8 V CMOS process.

3.4 Power Model

Power estimations are made using a flexible FPGA power model, called KPM, which is described in [16]. The power model, which is integrated into the VPR framework, estimates

dynamic, short-circuit, and static power of user circuits after they have been placed and routed onto a user-specified FPGA architecture. The model considers power dissipation within the logic blocks, routing fabric, and the clock distribution network of the FPGA.

KPM estimates power in two stages. In the first stage, the *transition density model* [27] is employed to determine the switching activity of each node in the user circuit. In the second stage, switching activities and transistor-level capacitance estimates are used to calculate the power dissipated by the user circuit. Like VPR, the model supports a wide variety of FPGA architectures and process technologies.

3.4.1 Switching Activity Estimation

Before calculating the dynamic and short-circuit power dissipation of a user circuit, the power model determines the switching activity of each node in the circuit. The switching activity estimates must be accurate in order to obtain accurate power estimates. Highly accurate switching activity estimation techniques, however, are computationally intensive and are not feasible for experiments involving many large circuits. Switching activity estimation techniques can be categorized into two groups: simulation-based and probabilistic-based.

Simulation based switching activity estimation techniques are typically more accurate than probabilistic techniques; however, they are more computationally intensive and they necessitate input vectors. There are a wide range of simulation based techniques, which simulate at different levels of abstraction. *Circuit level* simulators such as HSPICE offer very accurate results but they are only suitable for small circuits. *Gate level* simulators, which operate at the 0,1 abstraction level, are more suitable for large circuits.

Although less accurate, probabilistic based switching activity estimation techniques are significantly faster than simulation. There are many probabilistic techniques of varying complexity. The most accurate method is the transition density signal model.

The transition density of a signal is the expected number of times that the signal will toggle (from 1-to-0 or from 0-to-1) during each clock cycle. For a given signal, y, the transition density of the signal, $D(y)$, is determined using the following expression:

$$D(y) = \sum_{i=1}^{n} P(\frac{dy}{dx_i}) \cdot D(x_i)$$

where n is the number of signals that are input to function y, $P(dy/dx_i)$ is the probability that a change in x_i will cause a change in y, and $D(x_i)$ is the transition density of input x_i. The model assumes that inputs are spatially and temporally uncorrelated and it does not consider the inertial delay of logic gates. Because of these assumptions, switching activity may be severely overestimated in high-frequency circuits. To overcome this problem, a low-pass filter function [59] is applied to model the effect of gate delay on logic signals. Conceptually, this low-pass filter function prevents unrealistically short pulses from propagating.

3.4.2 Power Estimation

After estimating the switching activity of every node in the user circuit, KPM estimates overall FPGA power dissipation by considering the dynamic, short-circuit, and static power dissipated within its logic blocks, routing fabric, and clock distribution network. In the previous works [1, 2, 3, 4, 5, 6, 7], power estimates were obtained before placement and routing and only considered the dynamic power dissipated within the global routing fabric of the FPGA. Recent studies of FPGA power dissipation [16, 65] have reported that dynamic routing power accounts

for only half of the overall power. In order to properly evaluate power-aware FPGA CAD algorithms, a detailed power model that also considers the dynamic power dissipated within the logic blocks and the clock distribution network of the FPGA, as well as the overall static power dissipation must be employed.

Dynamic power is one of the main sources of FPGA power dissipation. It is dissipated whenever the node capacitances of an FPGA are charged or discharged. Dynamic power is expressed as follows:

$$Power_{dynamic} = 0.5 \cdot V_{supply} \cdot V_{swing} \cdot f_{clk} \cdot \sum_{i \in nodes} Activity(i) \cdot C_i$$

where V_{supply} is the supply voltage, V_{swing} is the swing voltage of each node, f_{clk} is the clock frequency of the circuit, and *Activity(i)* is the switching activity of node i with respect to f_{clk}, and C_i is the capacitance of node i. f_{clk} is determined using delay model, which calculates the critical-path delay of the user circuits.

KPM employs the transistor-level model from [60], the transistor sizing assumptions from [14], and wire length estimates based on architectural parameters to estimate the capacitance of embedded LUTs, multiplexers, and buffers within the logic blocks and the clock distribution network. It assumes the VPR logic block architecture, which is illustrated in Figure 3.3, and an H-tree clock distribution network, which is illustrated in Figure 3.4.

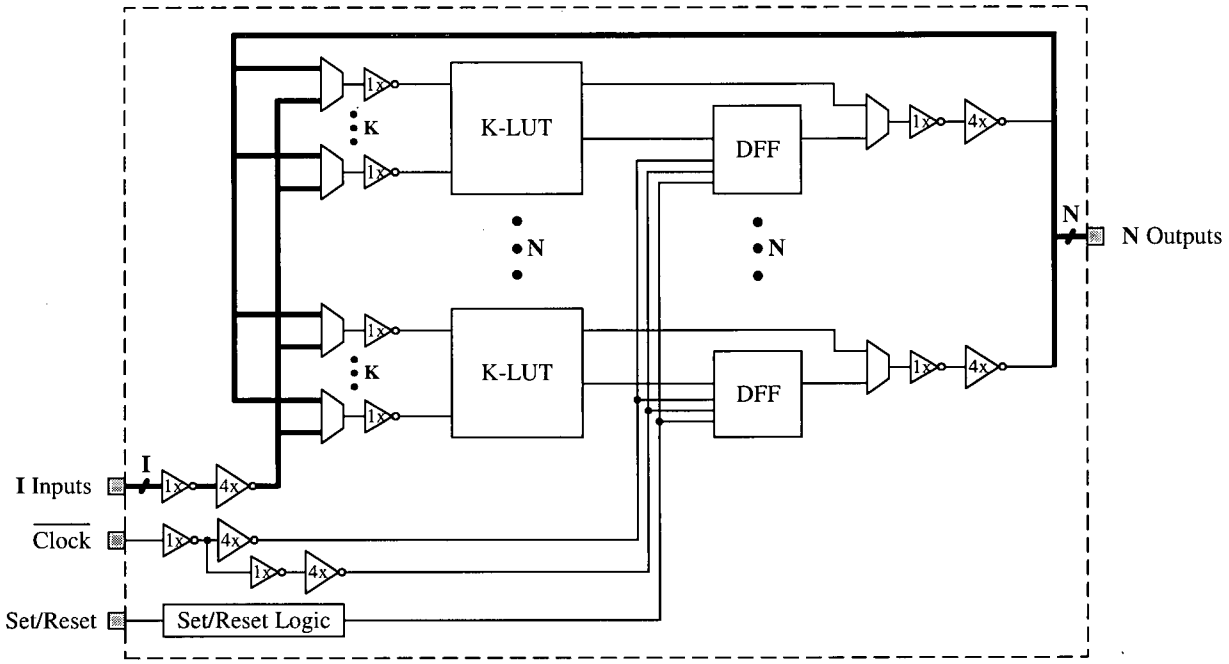


Figure 3.3: A generic VPR logic block.

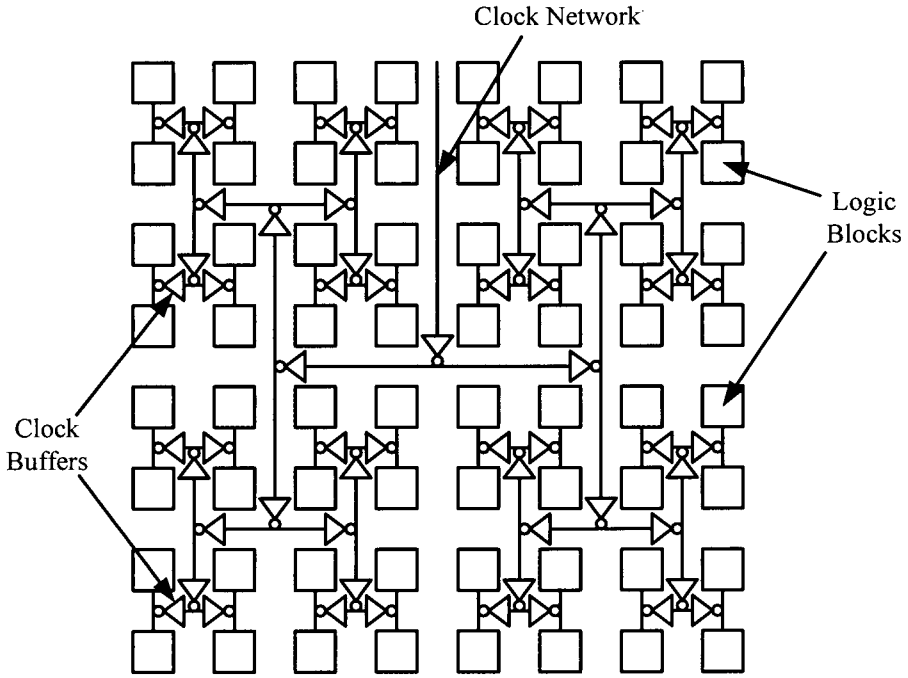


Figure 3.4: An H-tree clock distribution network.

Another component of power dissipation caused by signal switching is called *short-circuit* power. During a transition, there is a short period of time when the pull-up and pull-down networks of a static CMOS gate are "on" simultaneously. Power is dissipated during this period since current is allowed to flow from the supply rail to the ground rail. Short-circuit power is a function of the rise and fall time and the load capacitance [16]. Based on calculations using Altera and Xilinx datasheets [31, 61], KPM models short-circuit power as 10% of dynamic power.

The last component, called static power, is the power dissipated by transistors that are "off". Static power is caused by two types of transistor leakage current: drain leakage and subthreshold leakage. Both leakage currents are illustrated in Figure 3.5

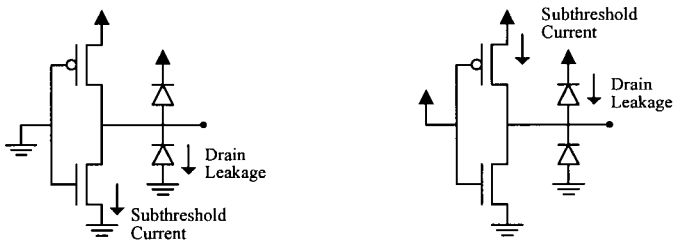


Figure 3.5: Leakage currents [62].

Drain leakage is current that flows through the reverse-biased diode junctions of the transistors located between the source or drain and the substrate. The main source of leakage, however, is subthreshold leakage, which is the current that flows between the source and the drain when the transistor is "off". KPM assumes that drain leakage is negligible and uses the following first-order estimation model to estimate the sub-threshold current [63]:

$$I_{drain}(weak\ inversion) = I_{on} \cdot \exp\left[\frac{(V_{gs} - V_{on}) \cdot q}{n \cdot k \cdot T}\right],$$

where:

- I_{on} is the drain current at the boundary when V_{gs} is equal to V_{on}
- V_{gs} is the gate-source voltage
- V_{on} is the boundary voltage between the weak and strong inversion regions
- q is the elementary charge
- n is a process parameter
- k is the Boltzman's constant
- T is the temperature in absolute temperature

KPM then calculates the static power dissipation of each transistor by multiplying the calculated subthreshold current with the supply voltage. KPM has been validated with HSPICE and has an average error of 13.4% [16]. Despite this significant absolute error, KPM was shown to have good fidelity; the relative comparisons between two alternative architectures or algorithms will be close to the relative errors that would be obtained by the actual devices and CAD tools.

3.5 Summary

This chapter described the experimental methodology employed to measure the individual and combined energy reductions of the power-aware algorithms that are presented in the following chapters. It described a methodology which compares the energy dissipated by circuits mapped onto an FPGA using power-aware algorithms to the energy dissipated by the same circuits when they are mapped using non power-aware algorithms. In contrast with the methodologies employed in previous works, power and delay estimations are obtained after placement and routing, and the power estimations include the static and dynamic power dissipated within the routing fabric, logic blocks, and clock distribution network of the FPGA.

Chapter 4

POWER-AWARE TECHNOLOGY MAPPING

This chapter begins by describing how power consumption can be minimized during the technology mapping stage of the FPGA CAD flow. In then describes EMap, a new power-aware technology mapping algorithm. Finally, it compares EMap to other published technology mapping algorithms.

4.1 Power and Technology Mapping

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, as shown in Figure 4.1.

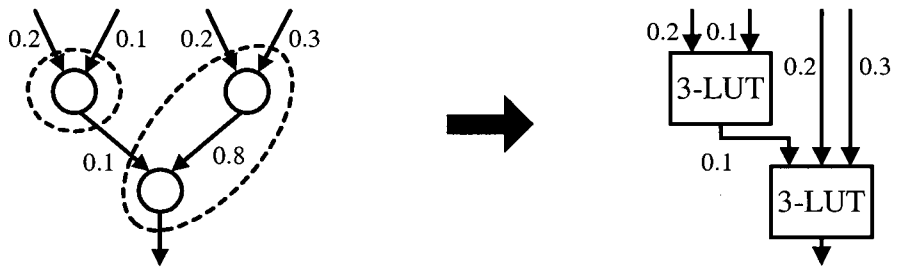


Figure 4.1: Activity-aware mapping solution.

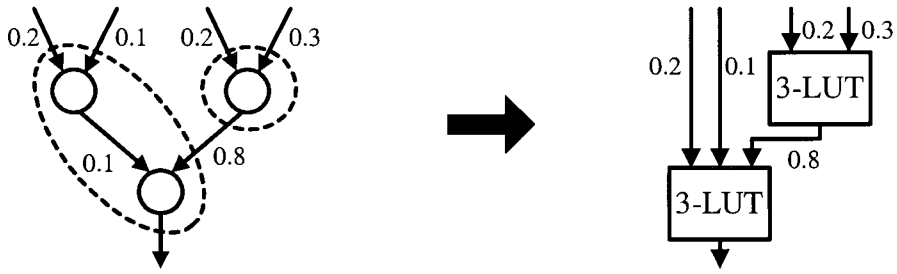


Figure 4.2: Non activity-aware mapping solution.

Figures 4.1 and 4.2 illustrate two different 3-LUT mapping solutions for the example Boolean network. The edges of the Boolean networks are annotated with switching activity values. By encapsulating the edge with the highest switching activity, the average activity of the edges of the 3-LUT network in Figure 4.1 is minimized. In Figure 4.2, however, the highest activity edge is not encapsulated and correspondly the average activity of the edges of the 3-LUT network is higher.

Another power reduction technique, recently described in [6], is to minimize node duplication, as described in Section 2.2.2. Technology mappers typically use node duplication to optimize for depth. However, node duplication increases the number of nodes and connections in an implementation, which increases the amount of power dissipated by the implementation. To demonstrate this, we compare two existing technology mappers that are not power-aware: FlowMap [67] and CutMap [11]. Both algorithms produce depth-optimal solutions. However, CutMap also attempts to minimize area by avoiding unnecessary node duplication. The results are shown in Table 4.1. On average, 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. Similar results are obtained for LUT sizes of 5 and 6.

Table 4.1: 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
5	FlowMap	2554	18.5	11301	11.9	2.53	11.9
	CutMap	2156	0	10102	0	2.26	0
6	FlowMap	2109	18.4	10179	11.6	2.59	12.1
	CutMap	1782	0	9118	0	2.31	0

4.2 EMap Algorithm

Our power-aware technology mapping algorithm, called EMap, incorporates the two techniques described above to reduce power. The EMap algorithm also minimizes the critical path delay by guaranteeing a depth-optimal solution. The algorithm has three phases.

4.2.1 Overview of the EMap Algorithm

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 [64]. 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. The algorithm is outlined in Figure 4.3.

```
/* 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 4.3: Pseudo-code of the EMap technology mapping algorithm.

4.2.2 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:

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

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 Experimental Methodology

To evaluate the influence of the technology-mapper on the energy dissipation of the circuits mapped onto FPGAs, we use the experimental framework described in Chapter 3. The experimental FPGA CAD flow is the same as the baseline FPGA CAD flow, except that CutMap is replaced with EMap. It is important to note that we take each benchmark circuit through the entire FPGA CAD flow, and then estimate power and delay. This is different than in previous works [1, 2, 3, 4, 5], 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 estimates.

4.4 Experimental Results

Table 4.2 summarizes the effect of the EMap technology mapping algorithm on the power, delay, and energy of each benchmark circuit. On average, the energy is reduced by 7.6%. 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 considers 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 [1, 2, 3, 4, 5], our improvement is approximately 21%.

Table 4.2: EMap results (K = 4).

Benchmark	CutMap	EMap	CutMap	EMap	CutMap	EMap
	Delay (ns)		Power (mW)		Energy (nJ)	
alu4	15.6	15.2	105	104	1.64	1.58
apex2	17.3	17.1	98.1	93.4	1.70	1.60
apex4	17.0	19.0	55.6	50.1	0.94	0.95
bigkey	8.4	8.4	280	298	2.34	2.51
clma	32.7	32.2	262	234	8.56	7.56
des	13.7	14.7	228	214	3.12	3.15
diffeq	17.9	17.7	50.3	44.7	0.90	0.79
dsip	10.1	9.1	219	242	2.22	2.21
elliptic	24.5	25.3	111	87.5	2.73	2.21
ex1010	24.2	23.7	118	108	2.85	2.56
ex5p	15.9	15.2	63.4	63.7	1.01	0.97
frisc	29.3	33.7	65.2	49.2	1.91	1.66
misex3	14.4	14.7	96	89.0	1.38	1.31
pdc	31.3	27.3	86	90.6	2.69	2.47
s298	28.3	30.5	78.9	68.9	2.23	2.10
s38417	20.6	20.1	361	332	7.43	6.69
s38584.1	23.4	16.5	252	299	5.89	4.92
seq	15.2	15.2	108	97.0	1.64	1.47
spla	22.7	24.9	88.9	74.7	2.02	1.86
tseng	18.2	18.8	53.0	44.4	0.96	0.84
Geo. Mean	18.9	18.7	115	107	2.18	2.01
% Diff	0.0	-0.75	0.0	-6.89	0.00	-7.59

Table 4.3 compares the energy dissipation of circuits mapped using FlowMap, CutMap, and EMap for various LUT sizes. As shown in the last column of the table, the energy improvement averaged over all benchmark circuits, is 7.6%, 8.4%, and 8.2% for LUT sizes of 4, 5, and 6, respectively.

Table 4.3: EMap 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 new technology mapper come primarily from the minimization of node duplication. As shown in columns 3 and 4 of Table 4.3, the EMap algorithm requires fewer nodes and connections than FlowMap and CutMap. The switching activity improvements account for only a small fraction of the gains. Figure 4.4 shows that 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 increase in the number of nodes and the number of connections more than counteracts the benefit of the activity reduction. The best results, shown in Table 4.2, where obtained when λ is set to 0.25.

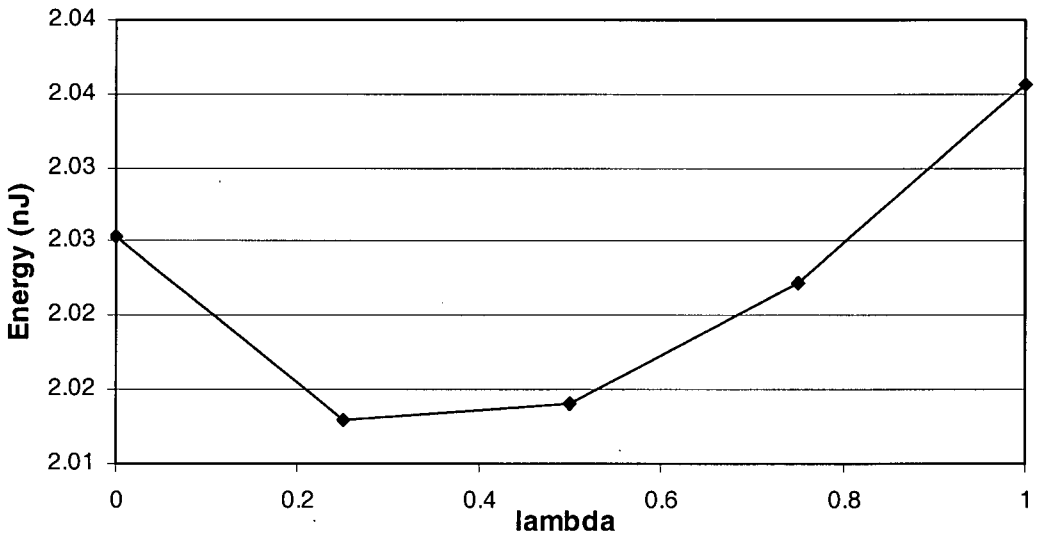


Figure 4.4: Energy versus the activity factor (lambda).

4.5 Summary

In this chapter, we described two methods of minimizing power dissipation during the technology mapping stage of the FPGA CAD flow. We then described EMap, a new depth-optimal power-aware technology mapping algorithm that reduces power by minimizing the amount of interconnect between LUTs and by hiding high activity signals within LUTs. Finally, using very detailed post-route power and delay models, we compared the performance of EMap to that of other published technology mappers. Specifically, we obtained an energy reduction of 7.6% and determined that most of the reductions were attributed to the minimization of node duplication. In the next chapter we use a similar approach to investigate power reduction during the clustering stage of the FPGA CAD flow.

Chapter 5

POWER-AWARE CLUSTERING

We begin this chapter by describing how energy dissipation can be minimized during the clustering stage of the FPGA CAD flow. We then describe a new power-aware clustering algorithm called P-T-VPack, which is a straightforward enhancement of the T-VPack clustering algorithm. Finally, we compare the energy dissipation of circuits clustered using P-T-VPack to that of circuits clustered using T-VPack.

5.1 Energy and Clustering

Minimizing energy during the clustering stage of the FPGA CAD flow is similar to minimizing energy during technology mapping. The goal is to encapsulate high activity wires within clusters, where the connections dissipate less energy. Intuitively, we would expect clustering to be more effective than technology mapping at reducing power, since clusters are typically larger than LUTs (commercial parts contain as many as 10 LEs per cluster). On the other hand, encapsulating high activity connections within clusters does not eliminate these connections 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. Figure 5.1 illustrates an inter-cluster and an intra-cluster connection. Using KPM with the benchmark circuits and the FPGA

architecture described in Section 3.2, the average capacitance of intra-cluster and inter-cluster connections were found to be 25 fF and 450 fF, respectively.

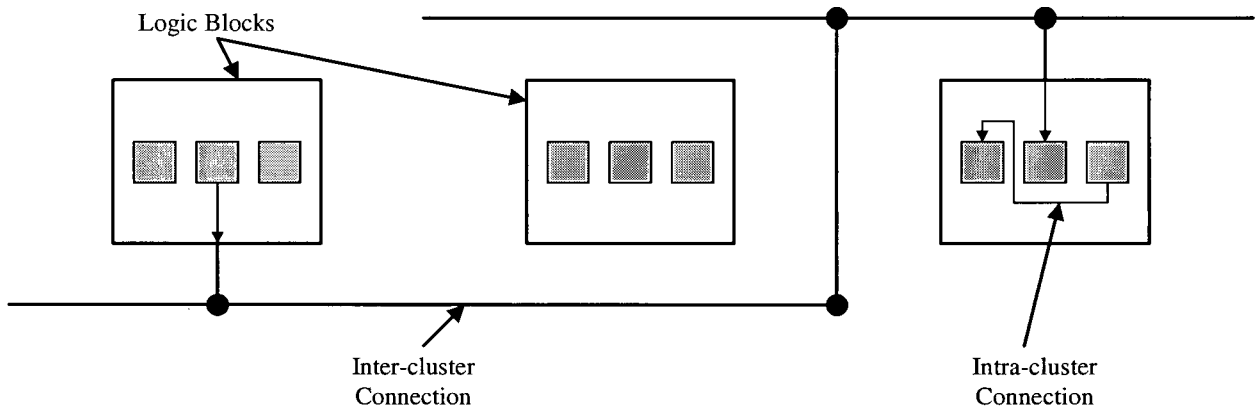


Figure 5.1: Inter-cluster and Intra-cluster connections.

To investigate the tradeoffs outlined above, we enhance the T-VPack algorithm, described in Section 2.2.3, to minimize energy. The T-VPack algorithm is enhanced in two ways. The first enhancement is to recalibrate the original T-VPack algorithm to minimize energy. The second is to modify the attraction functions employed by T-VPack to consider the switching activity of connections between LEs. These enhancements are described in the following two subsections.

5.2 Recalibrating the T-VPack Algorithm

The T-VPack algorithm simultaneously minimizes the amount of interconnect between clusters and the critical-path delay of user circuits, as described in Section 2.2.3. In T-VPack, the relative importance of the interconnect and delay minimization objectives is controlled with a timing-tradeoff parameter, denoted α. The T-VPack algorithm was originally calibrated to minimize the critical-path delay of user circuits. A default α value of 0.75 was shown to produce fast circuits. However, since minimal delay does not imply minimal energy, we begin our enhancement of T-VPack by recalibrating the algorithm to minimize energy. Specifically,

we find the value for the timing-tradeoff parameter, α, of the T-VPack attraction function that produces the lowest energy results. The results are shown in Figure 5.2.

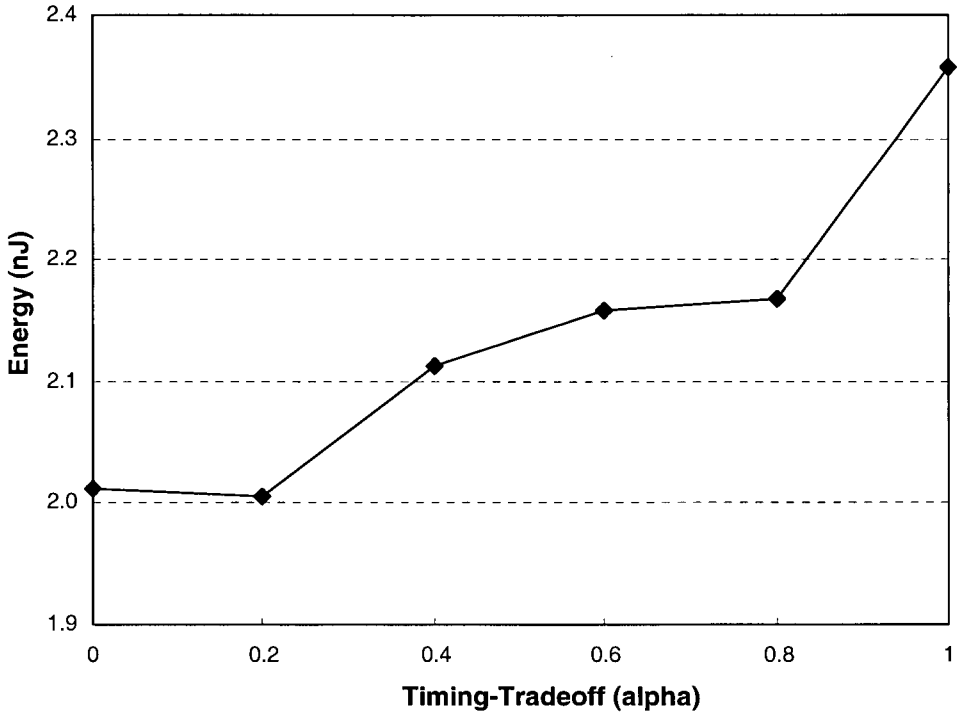


Figure 5.2: Energy versus timing-tradeoff.

Figure 5.2 is a plot of energy with respect to the timing-tradeoff parameter, α. When α is 0, the clustering algorithm focuses entirely on minimizing the amount of interconnect between clusters. When α is 1, the clustering algorithm focuses entirely on minimizing the critical-path delay. The results indicate that the default α value of 0.75 is not good in terms of energy. The lowest energy results are obtained when α is 0.2. When α is changed to 0.2 instead of 0.75 (default value), the energy dissipation is reduced by 7.0%.

5.3 Enhancing the T-VPack Algorithm

In the previous subsection, we recalibrated the baseline clustering algorithm for energy by changing the timing-tradeoff parameter, α. In this subsection, we enhance the baseline

clustering algorithm for energy by modifying the algorithm's attraction functions to consider the switching activity of connections while packing clusters.

The first attraction function, which selects the first LE that is packed into each cluster, is modified to select the LE whose input and output wires have high switching activities. By initializing a cluster with an LE that has high activity input and output wires, high activity wires are more likely to be encapsulated within the cluster.

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) :

$$Attraction(B) = \alpha \cdot Crit(B) + (1-\alpha) \cdot \left[(1-\beta) \cdot \frac{\sum Weight(i) \mid i \in Nets(B) \cap Nets(C)}{G} + \beta \cdot \frac{\sum Activity(i) \mid i \in Nets(B) \cap Nets(C)}{G \cdot Activity_{Avg}} \right]$$

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, $Activity_{Avg}$ 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.

5.4 Experimental Methodology

To compare the influence of the power-aware clustering algorithm on the energy dissipation, we use the experimental framework described in Chapter 3, with T-VPack replaced with P-T-VPack. In both cases, the baseline technology mapper, placer, and router were used. Again, each benchmark circuit is passed through the entire FPGA CAD flow before the estimating the power and critical-path delay of each circuit.

5.5 Experimental Results

The enhanced algorithm, called P-T-VPack, has two parameters, α and β, which control the relative importance of each term in the second attraction function. Before measuring the energy reductions achieved by the new algorithm, we experimentally determine the value of each parameter that results in the lowest overall energy.

5.5.1 Calibrating the P-T-VPack Algorithm

Figure 5.3 is a plot of energy with respect to the timing-tradeoff parameter, α, and the power-tradeoff parameter, β. When α is 0, the attraction function focuses entirely on the wiring

component of the attraction function, and when α is 1, it focuses entirely on minimizing critical-path delay. Similarly, when β is 0, the wiring component of the attraction function focuses on minimizing the amount of interconnect between clusters, and when β is 1, it focuses on minimizing the switching activity of interconnect between clusters. The results indicate that the lowest energy results are obtained when α is 0.0 and β is between 0.4 and 0.8.

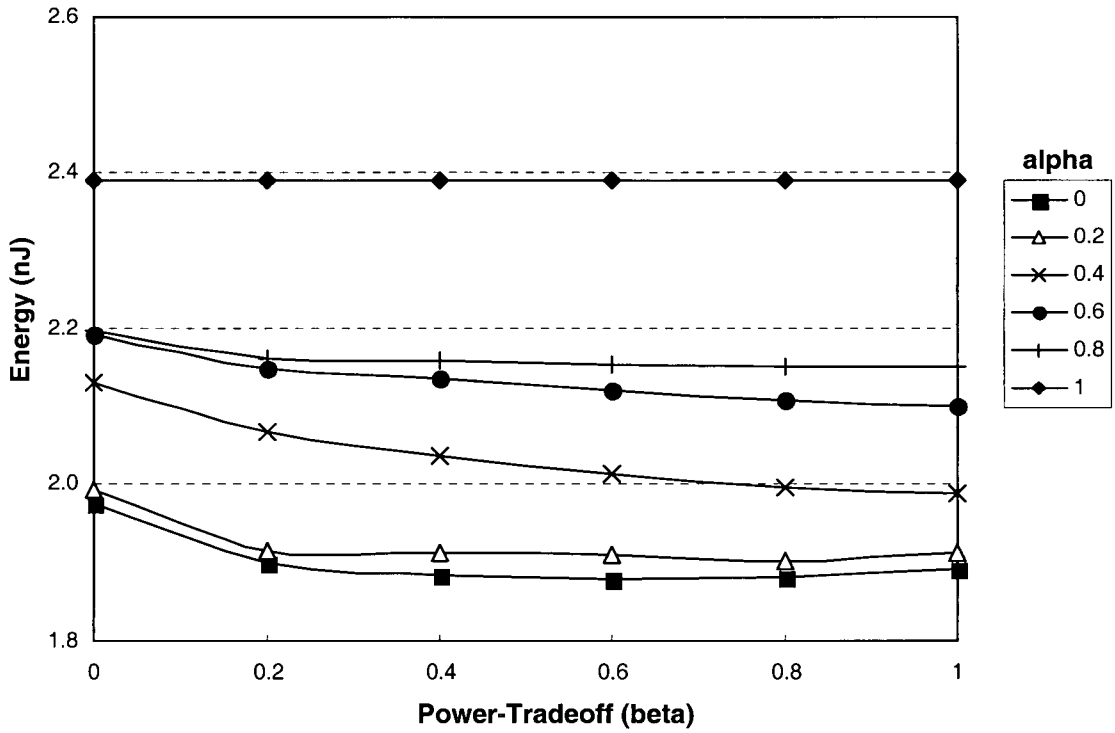


Figure 5.3: Energy versus alpha and beta.

5.5.2 Final Results

Table 5.1 summarizes the effect of the power-aware clustering algorithm on the power, delay, and energy of each benchmark circuit. On average, the energy is reduced by 12.6% (5.6% more than the recalibrated T-VPack algorithm).

Table 5.1: P-T-VPack results (N = 4).

Benchmark	T-VPack	P-T-VPack	T-VPack	P-T-VPack	T-VPack	P-T-VPack
	Delay (ns)		Power (mW)		Energy (nJ)	
alu4	15.6	16.2	105	96.3	1.64	1.56
apex2	17.3	19.1	98.1	78.4	1.70	1.50
apex4	17.0	16.0	55.6	51.9	0.94	0.83
bigkey	8.4	9.0	280	267	2.34	2.40
clma	32.7	37.4	262	198	8.56	7.39
des	13.7	15.8	228	198	3.12	3.13
diffeq	17.9	19.5	50.3	38.7	0.90	0.76
dsip	10.1	8.0	219	255	2.22	2.04
elliptic	24.5	25.2	111	82.0	2.73	2.07
ex1010	24.2	24.3	118	101	2.85	2.46
ex5p	15.9	15.6	63.4	56.0	1.01	0.87
frisc	29.3	34.3	65.2	42.6	1.91	1.46
misex3	14.4	16.1	96	77.1	1.38	1.24
pdc	31.3	27.9	86	83.3	2.69	2.32
s298	28.3	29.4	78.9	60.0	2.23	1.76
s38417	20.6	29.1	361	227	7.43	6.60
s38584.1	23.4	18.4	252	282	5.89	5.19
seq	15.2	15.1	108	92.6	1.64	1.40
spla	22.7	30.6	88.9	56.4	2.02	1.73
tseng	18.2	18.1	53.0	46.8	0.96	0.85
Geo. Mean	18.9	19.7	115	96.4	2.18	1.90
% Diff	0.0	4.48	0.0	-16.4	0.00	-12.6

Figure 5.4 compares the energy reductions of the P-T-VPack algorithm for different clusters sizes. 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.

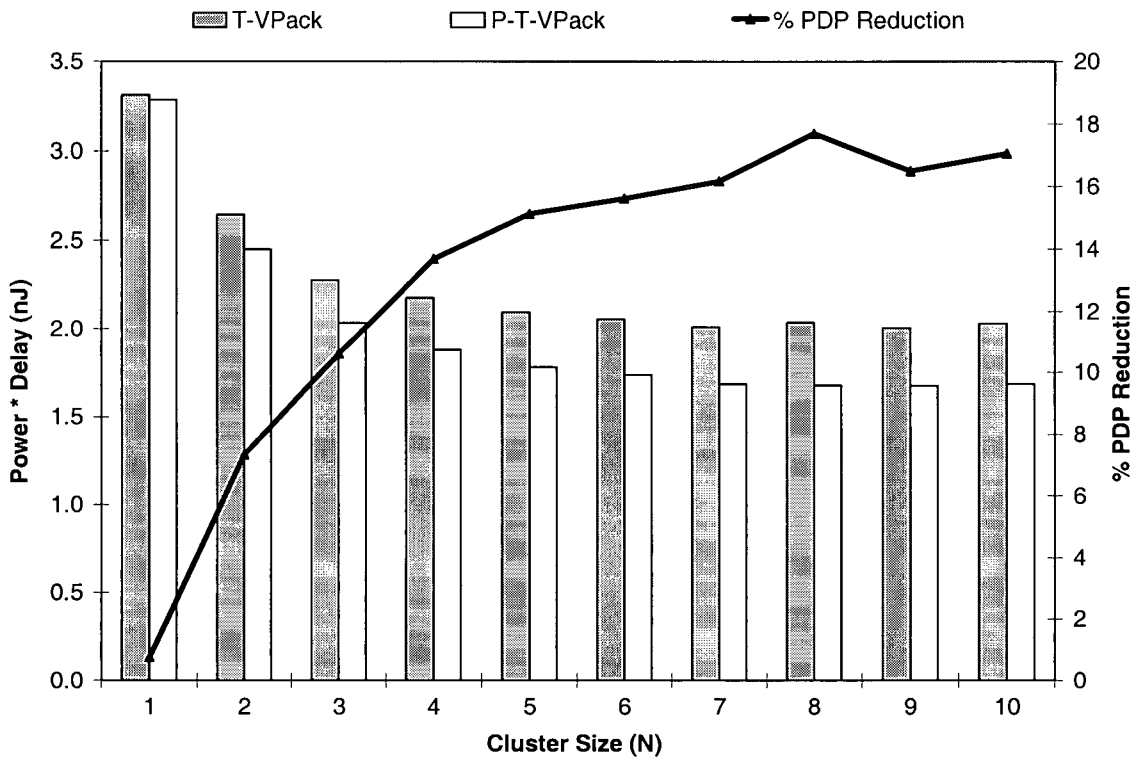


Figure 5.4: Energy dissipation versus cluster size.

Finally, in Table 5.2 we examine the gains in more detail. For clusters with four LEs, the power-aware packer 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 5.2: Clustering gain components.

	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

5.6 Summary

In this chapter, we described two methods of reducing energy dissipation during the clustering stage of the FPGA CAD flow. We then described P-T-VPack, a new power-aware clustering

algorithm that is based on the T-VPack algorithms. The P-T-VPack algorithm reduces power by minimizing the amount of interconnect between logic blocks and by hiding high-activity signals within logic blocks. Finally, we compared the energy dissipation of circuits clustered using P-T-VPack to that of circuits clustered using T-VPack. Our experimental results indicate that clustering is well suited for power optimization, with average energy reductions of 12.6% for clusters with four LEs. In the next chapter, we investigate energy minimization during the placement stage of the FPGA CAD flow.

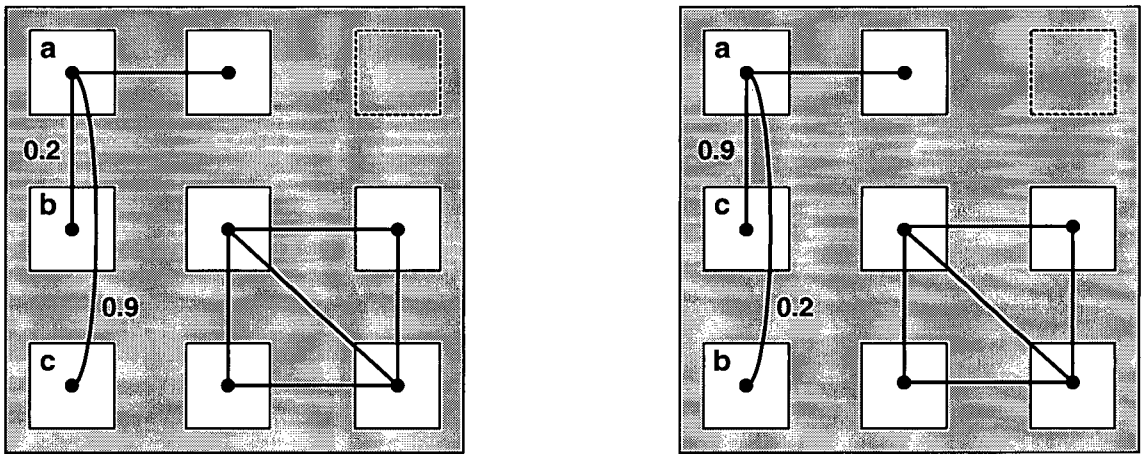
Chapter 6

POWER-AWARE PLACEMENT

We begin this chapter by describing how energy dissipation can be minimized during the placement stage of the FPGA CAD flow. We then describe how we enhanced our baseline placement algorithm to be power-aware. Finally, we compare the performance of the enhanced algorithm to that of the baseline algorithm, T-VPlace.

6.1 Energy and 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.



a) Placement solution 1 b) Placement solution 2

Figure 6.1: Two example placement solutions.

Figure 6.1 illustrates two possible placement solutions for the same user circuit. The connection between logic blocks (a) and (b) has a switching activity of 0.2, while the connection between logic blocks (a) and (c) has a switching activity of 0.9. In the first placement solution, (b) is adjacent to (a), and in the second placement solution, (c) is adjacent to (a). The second placement solution is better in terms of power since the high activity connection between (a) and (c) is likely to be shorter 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 high-activty 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 reductions, thereby reducing the overall energy reductions. To investigate these tradeoffs, we enhanced an existing timing-aware placement algorithm, called T-VPlace, to minimize energy.

6.2 Recalibrating the T-VPlace Algorithm

The T-VPlace algorithm, described in section 2.2.4, simultaneously minimizes the overall wire length and the critical-path delay of user circuits during the placement stage of the FPGA CAD flow. A user-defined timing-tradeoff parameter, denoted λ, determines the relative importance of the wiring and delay minimization objectives. The T-VPlace algorithm was originally calibrated to minimize circuit delay, not energy. Therefore, we begin our enhancement of T-VPlace by recalibrating the algorithm to minimize energy. Specifically, we find the value for λ, which produces the lowest energy placements. The results are shown in Figure 6.2.

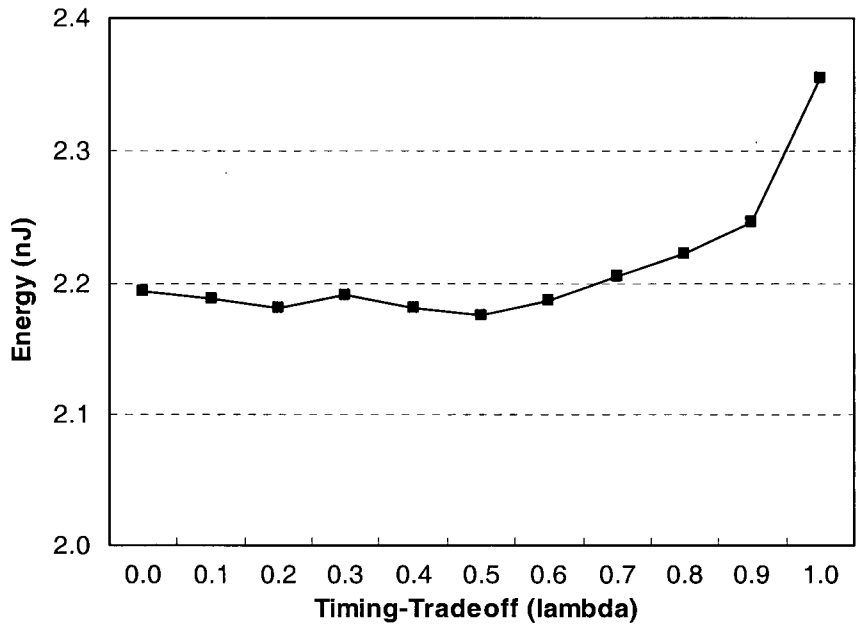


Figure 6.2: Energy versus timing-tradeoff.

Figure 6.2 is a plot of energy with respect to the timing-tradeoff parameter, λ. When λ is 0, the placement algorithm focuses entirely on minimizing wire length. When λ is 1, the placement focuses entirely on minimizing critical-path delay. The results indicate that the default λ value of 0.5 is good in terms of energy; the results are only slightly improved (by 0.1%) when λ is 0.2.

6.3 Enhancing the T-VPlace Algorithm

As described above, the energy reductions of the original algorithm can also be improved by considering the switching activity of the nets during placement. Signals that toggle more often consume more power. By placing logic blocks that are connected by high activity nets closer together, the energy dissipated within the global routing fabric may be reduced. To make the algorithm activity-aware, we modify the cost function of the original algorithm as follows:

$$\Delta C = \lambda \cdot \frac{\Delta TimingCost}{PreviousTimingCost} + (1-\lambda) \cdot \left[(1-\gamma) \cdot \frac{\Delta WiringCost}{PreviousWiringCost} + \gamma \cdot \frac{\Delta PowerCost}{PreviousPowerCost} \right]$$

The timing and wiring cost component are the same as before (see section 2.2.4). However, as described in [8], a power cost component is added to the cost function and a power-tradeoff parameter, γ, is added to control the relative importance of the power cost with respect to the wiring cost. The power cost component estimates the power consumption of each net by multiplying their bounding box and switching activity:

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

where the bounding box term, $q(i) \cdot [bb_x(i) + bb_y(i)]$, is the same as before and estimates the capacitance of net i, and $Activity(i)$ estimates the switching activity of net 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.

6.4 Experimental Methodology

To compare the influence of the power-aware placement algorithm on energy dissipation, we use the experimental framework described in Chapter 3, with T-VPlace replaced with P-T-VPlace. Each benchmark circuit is passed through the entire FPGA CAD flow before the estimating the power and critical-path delay of each circuit. The results are then compared to those obtained using the baseline the CAD flow.

6.5 Experimental Results

The enhanced algorithm, called P-T-VPlace, has two parameters, λ and γ, which control the relative importance of each term of the cost function. Before measuring the energy reductions of the enhanced algorithm, we determine the lowest energy value for each parameter experimentally.

6.5.1 Calibrating the P-T-VPlace Algorithm

Figure 6.2 is a plot of energy with respect to the timing-tradeoff parameter, λ, and the power-tradeoff parameter, γ. When λ is 0, the attraction function focuses entirely on the wiring component of the attraction function, and when λ is 1, it focuses entirely on minimizing critical-path delay. Similarly, when γ is 0, the wiring component of the attraction function focuses on minimizing the amount of interconnect between clusters, and when γ is 1, it focuses on minimizing the switching activity of interconnect between clusters. The results indicate that setting λ to 0.2 and γ to 0.8 is the best in terms of energy.

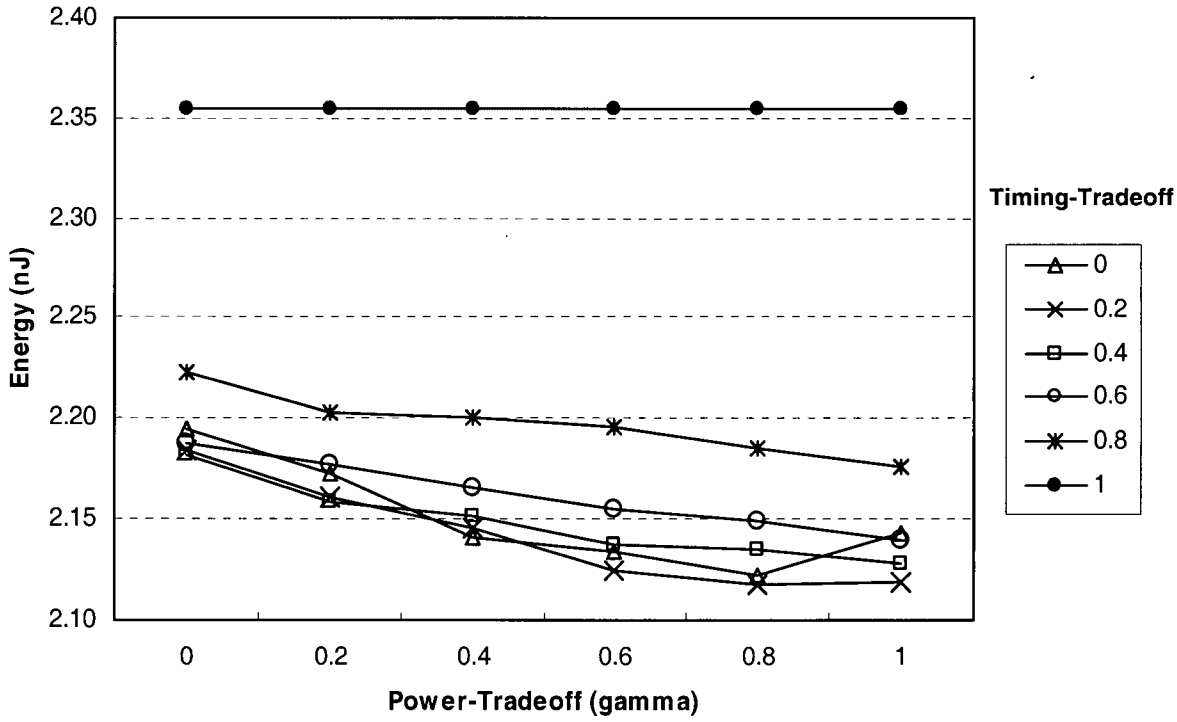


Figure 6.3: Energy versus power-tradeoff (gamma).

6.5.2 Final Results

P-T-VPlace produces marginal but consistent improvements in terms of energy when compared with T-VPlace, as shown in Table 6.1. On average the power-aware algorithm reduces energy by 3.0%, where all benchmarks showed improvement. 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.

Table 6.1: P-T-VPlace results.

Benchmark	T-VPlace	P-T-VPlace	T-VPlace	P-T-VPlace	T-VPlace	P-T-VPlace
	Delay (ns)		Power (mW)		Energy (nJ)	
alu4	15.6	16.3	105	99.6	1.64	1.62
apex2	17.3	19.3	98.1	85.4	1.70	1.64
apex4	17.0	19.9	55.6	44.9	0.94	0.90
bigkey	8.4	8.8	280	264	2.34	2.33
clma	32.7	36.2	262	225	8.56	8.15
des	13.7	14.6	228	212	3.12	3.08
diffeq	17.9	19.2	50.3	45.9	0.90	0.88
dsip	10.1	8.5	219	258	2.22	2.19
elliptic	24.5	24.7	111	106	2.73	2.61
ex1010	24.2	23.7	118	119	2.85	2.81
ex5p	15.9	16.5	63.4	59.4	1.01	0.98
frisc	29.3	33.3	65.2	52.8	1.91	1.76
misex3	14.4	16.2	96	84.6	1.38	1.37
pdc	31.3	32.9	86	77.2	2.69	2.54
s298	28.3	30.0	78.9	72.3	2.23	2.17
s38417	20.6	21.6	361	336	7.43	7.25
s38584.1	23.4	15.3	252	380	5.89	5.82
seq	15.2	15.0	108	106	1.64	1.59
spla	22.7	32.1	88.9	59.4	2.02	1.91
tseng	18.2	18.5	53.0	50.9	0.96	0.94
Geo. Mean	18.9	19.6	115	108	2.18	2.11
% Diff	0.0	3.96	0.0	-6.68	0.00	-2.99

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 for both the baseline and the power-aware placement algorithms. The total capacitance of all the nets in each group was then summed, and the results are plotted in Figure 6.4. 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.

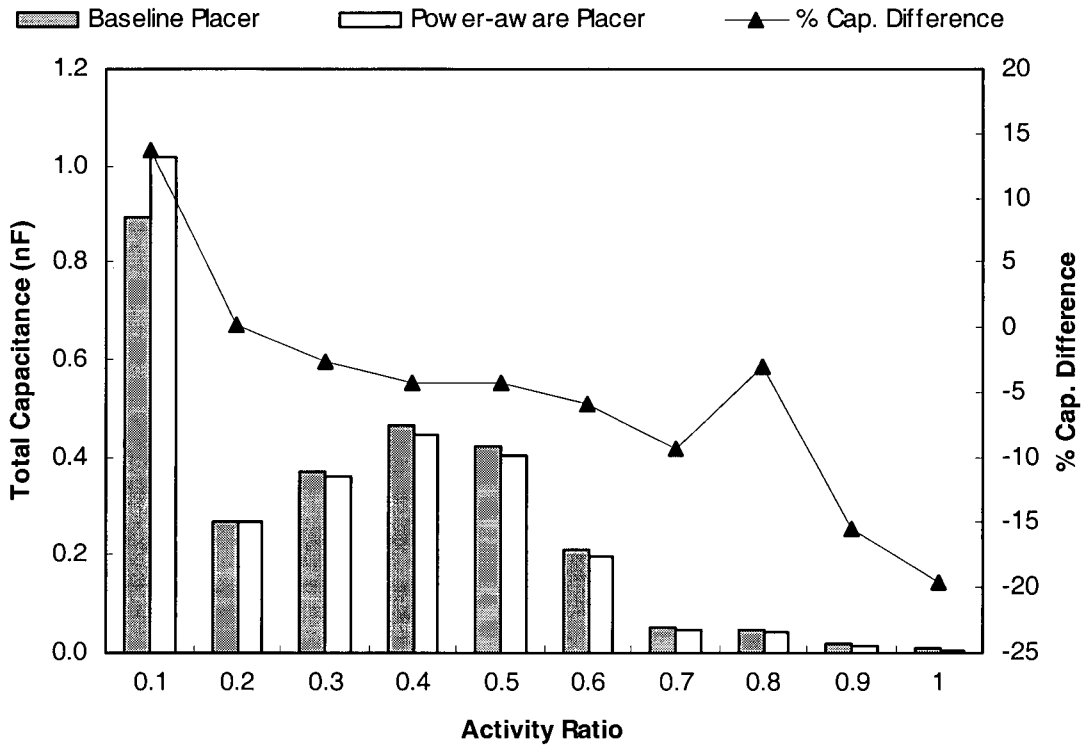


Figure 6.4: P-T-VPlace (wire cap. vs. switching activity).

6.6 Summary

In this chapter we described how to reduce power consumption during the placement stage of the FPGA CAD flow. We then described P-T-VPlace, a power-aware placement algorithm that is based on the T-VPlace algorithm. P-T-VPlace reduces power by placing logic blocks that are connected by nets with high switching-activity close together. Finally, we compared the energy dissipation of circuits placed using P-T-VPlace to that of circuits placed using T-VPlace. Our experimental results indicate that placement is not as well suited for energy minimization as are technology mapping and clustering, with energy reductions of only 3.0%. We found that much of the power reductions achieved by the power-aware algorithm were counteracted by an increase in the critical-path delay. In the next chapter we investigate power reduction during the routing stage of the FPGA CAD flow.

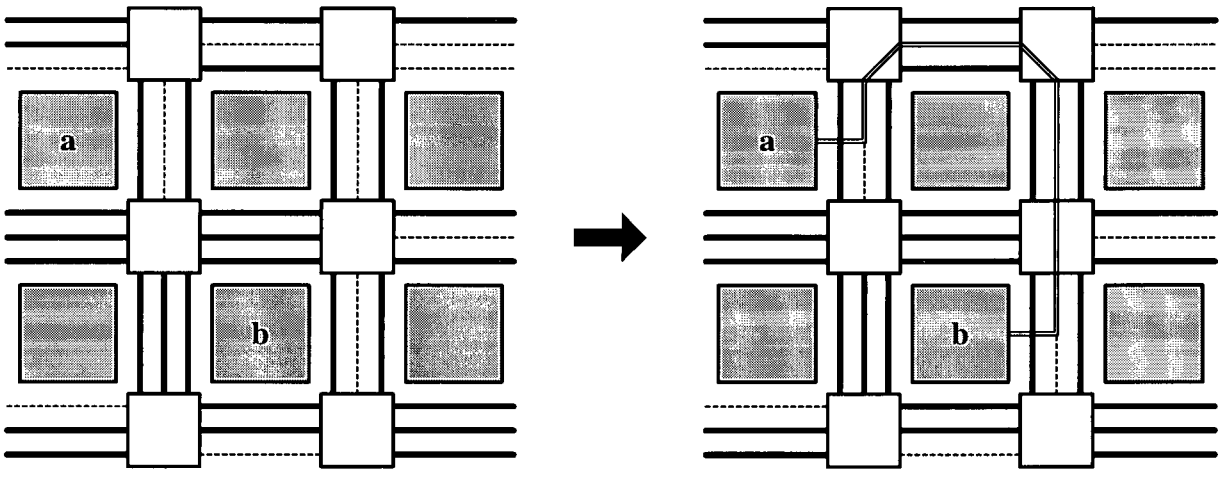
Chapter 7

POWER-AWARE ROUTING

This chapter begins with a description of how energy dissipation can be minimized during the routing stage of the FPGA CAD flow. It then describes how the VPR routing algorithm was enhanced to minimize energy. Finally, it compares the energy dissipation of circuits routed using the power-aware VPR router to that of circuits routed using the original VPR router.

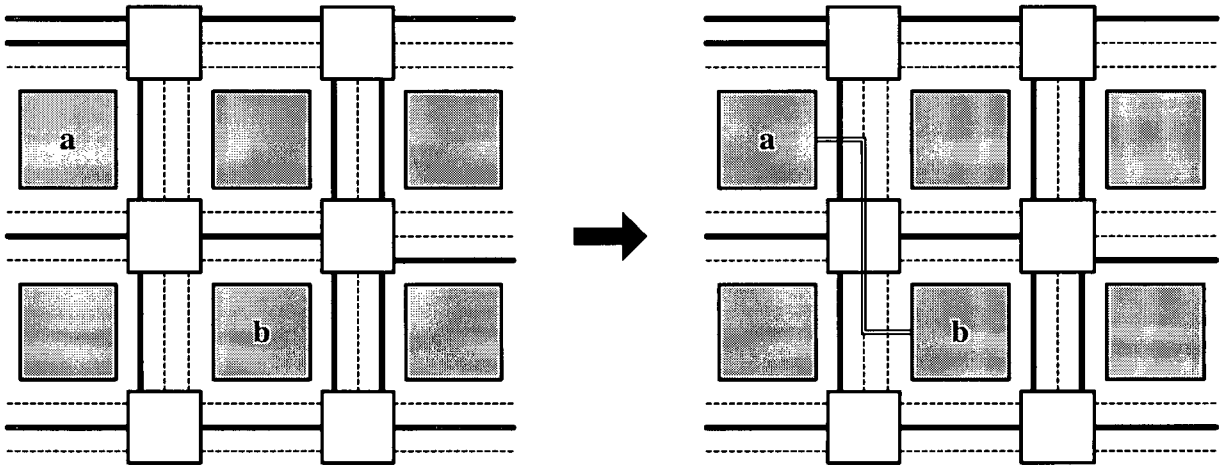
7.1 Energy and Routing

The ideal routing solution, in terms of energy, is one where each net is routed as directly as possible. In practice, however, it is not possible to route every net directly since FPGA routing channels have a limited number of tracks. When routing channels become congested, some connections must use alternative routes that are less direct, as shown in Figure 7.1.



a) Congested interconnect b) Indirect connection

Figure 7.1: Routing example 1: (a) congested interconnect and (b) indirect connection.



a) Uncongested interconnect b) Direct connection

Figure 7.2: Routing example 2: (a) uncongested interconnect and (b) direct connection.

In Figure 7.1 and Figure 7.2, the dark routing tracks are occupied, the dotted routing tracks are not occupied, and the double lined routing tracks indicate the path taken to route the connection between logic block *a* and logic block *b*. In Figure 7.1, the connection between *a* and *b* is routed indirectly because the channels between the logic blocks are congested. In Figure 7.2, however, the connection can be routed directly since the channels are not congested. The indirect connection requires four routing wires and three switch blocks, whereas, the direct connection requires only two routing wires and one switch block. Correspondingly, the indirect connection dissipates significantly more energy than a direct connection.

Although it is not possible to route every net directly, the energy dissipated within the routing fabric of the FPGA may be reduced by routing high activity nets more directly than low activity nets. Intuitively, we would expect the energy reductions obtained from such a power-aware router to be similar to those obtained from the power-aware placement algorithm described in Chapter 6. In both cases, the power-aware algorithm can not eliminate high activity nets all together. Instead, they can only make the high activity nets shorter. Also, in both cases, there is

a tradeoff between power and delay. When there are timing-critical nets that have low switching activity, the delay of the circuit may increase since the low switching activity of the net reduces the net's priority. This delay increase may counteract the power reductions, thereby reducing the overall energy reductions. To investigate these issues, we enhanced the timing-aware VPR routing algorithm that is described in section 2.2.5.

7.2 The P-T-VRoute Algorithm

The new enhanced routing algorithm, called P-T-VRoute, is nearly the same as the timing-aware VPR routing algorithm. Only the cost function of the algorithm is modified to consider the switching activity of the nets, as follows:

$$Cost(n) = Crit(i,j) \cdot delay_{Elmore}(n) + (1 - Crit(i,j)) \cdot [ActCrit(i) \cdot k \cdot cap(n) + (1 - ActCrit(i)) \cdot b(n) \cdot h(n) \cdot p(n)]$$

where $Crit(i,j)$, $delay_{Elmore}(n)$, $b(n)$, $h(n)$, and $p(n)$ are the same as before (see section 2.2.5), k is a power-tradeoff parameter which determines the relative importance of the new power term, $cap(n)$ is the normalized capacitance associated with routing resource node n, and $ActCrit(i)$ is the activity criticality. The activity criticality of net i is defined as:

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

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 is permitted to have. Setting $MaxActCrit$ to 0.99 prevents nets with very high activity from completely ignoring congestion.

The delay term of the cost function is left unchanged; when the timing criticality of a connection is high, the router focuses on minimizing the Elmore delay of the connection. The second term of the cost function, however, is modified to consider switching activity. When the switching activity of a net is high, the power-aware router searches for the path from the source to the sink of the net with the lowest capacitance. Otherwise, the router behaves as it did before, searching for low congestion paths.

7.3 Experimental Methodology

Again, we use the experimental framework described in Chapter 3, with T-VRoute replaced with P-T-VRoute, to compare the influence of the power-aware routing algorithm on energy dissipation. Each benchmark circuit is passed through the entire FPGA CAD flow before estimating the power and critical-path delay of each circuit. The results are then compared to those obtained using the baseline CAD flow.

7.4 Experimental Results

The P-T-VRoute algorithm has a power-tradeoff parameter, k, which controls the relative importance of the power term in the cost function. Before measuring the energy reductions of the enhanced algorithm, we experimentally determine the best value for this parameter.

7.4.1 Calibrating the P-T-VRoute Algorithm

Figure 7.3 is a plot of the power, delay, and energy reductions of the power-aware router with respect to the power-tradeoff parameter, k. When k is 0, the algorithm does not consider switching activity. As k increases, however, the algorithm increasingly focuses on minimizing the capacitance of high activity nets. Figure 7.3 clearly illustrates the relationship between

power and delay. As k increases, power consumption decreases but critical-path delay increases. The delay increase counteracts most of the energy reductions. The power-aware router achieves the maximum energy reductions (3.0%) when k is 3; however, the critical-path delay when k is 3 increases significantly (8.2%). On the other hand, the power-aware router achieves comparable energy reductions (2.7%) when k is 1.5, while only increasing the critical-path delay by 3.8%. Therefore, the default k value is set to 1.5.

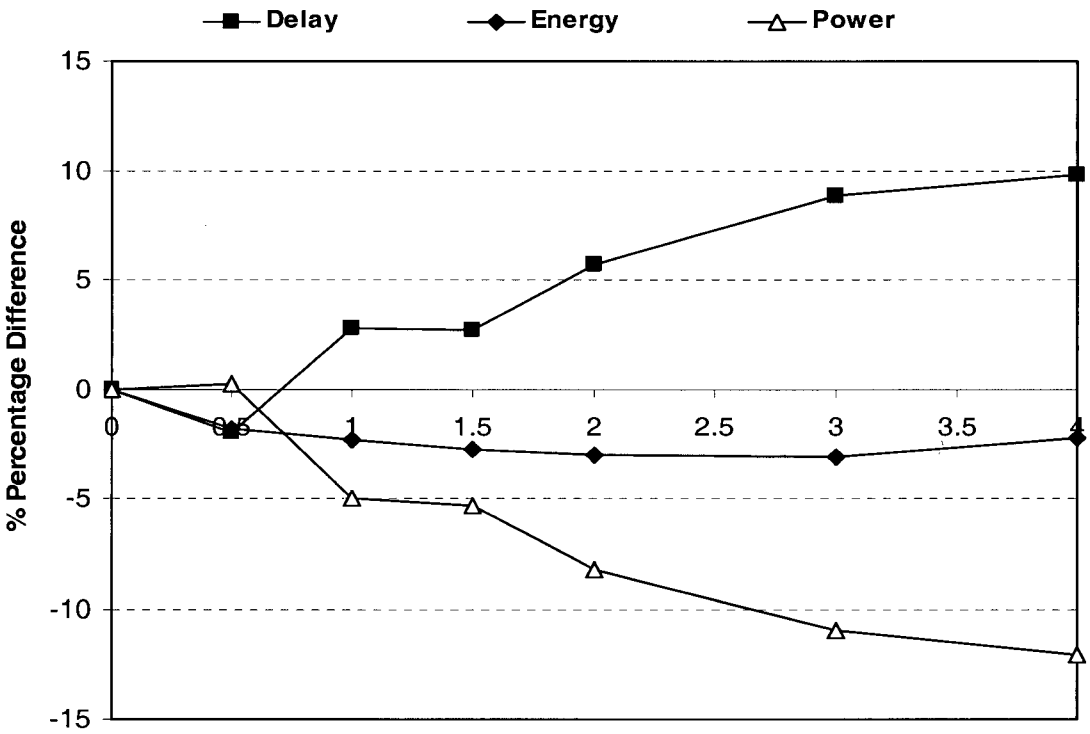


Figure 7.3: Energy, Delay, and Power versus Power Weight.

7.4.2 Experimental Results

The energy reductions of the power-router were similar to those achieved during placement. The average energy dissipation was reduced by 2.7%, where all of the 20 benchmarks showed an improvement. The power-aware router reduces global interconnect power by 6.2% compared to the baseline placer. The critical-path delay, however, increases by 3.8%, thereby counteracting much of the power reductions. Again, 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.

Table 7.1: P-T-VRoute results.

Benchmark	T-VRoute	P-T-VRoute	T-VRoute	P-T-VRoute	T-VRoute	P-T-VRoute
	Delay (ns)		Power (mW)		Energy (nJ)	
alu4	15.6	15.3	105	103	1.64	1.58
apex2	17.3	17.2	98.1	95.1	1.70	1.64
apex4	17.0	15.3	55.6	60.5	0.94	0.93
bigkey	8.4	8.7	280	263	2.34	2.29
clma	32.7	32.9	262	248	8.56	8.15
des	13.7	14.4	228	212	3.12	3.05
diffeq	17.9	18.4	50.3	47.4	0.90	0.87
dsip	10.1	19.3	219	114	2.22	2.21
elliptic	24.5	23.4	111	113	2.73	2.63
ex1010	24.2	24.7	118	113	2.85	2.78
ex5p	15.9	15.8	63.4	62.4	1.01	0.98
frisc	29.3	31.4	65.2	59.2	1.91	1.86
misex3	14.4	14.3	96	96.0	1.38	1.37
pdc	31.3	28.1	86	93.2	2.69	2.62
s298	28.3	31.3	78.9	68.7	2.23	2.15
s38417	20.6	33.3	361	218	7.43	7.27
s38584.1	23.4	16.0	252	359	5.89	5.76
seq	15.2	14.7	108	107	1.64	1.58
spla	22.7	22.6	88.9	86.6	2.02	1.96
tseng	18.2	18.2	53.0	51.8	0.96	0.94
Geo. Mean	18.9	19.6	115	108	2.18	2.12
% Diff	0.0	3.77	0.0	-6.17	0.00	-2.63

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, the same technique described in section 6.5.2 was used to examine the relationship between wire capacitance and switching activity of nets routed with the power-aware router. Again, as shown in Figure 7.4, the 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.

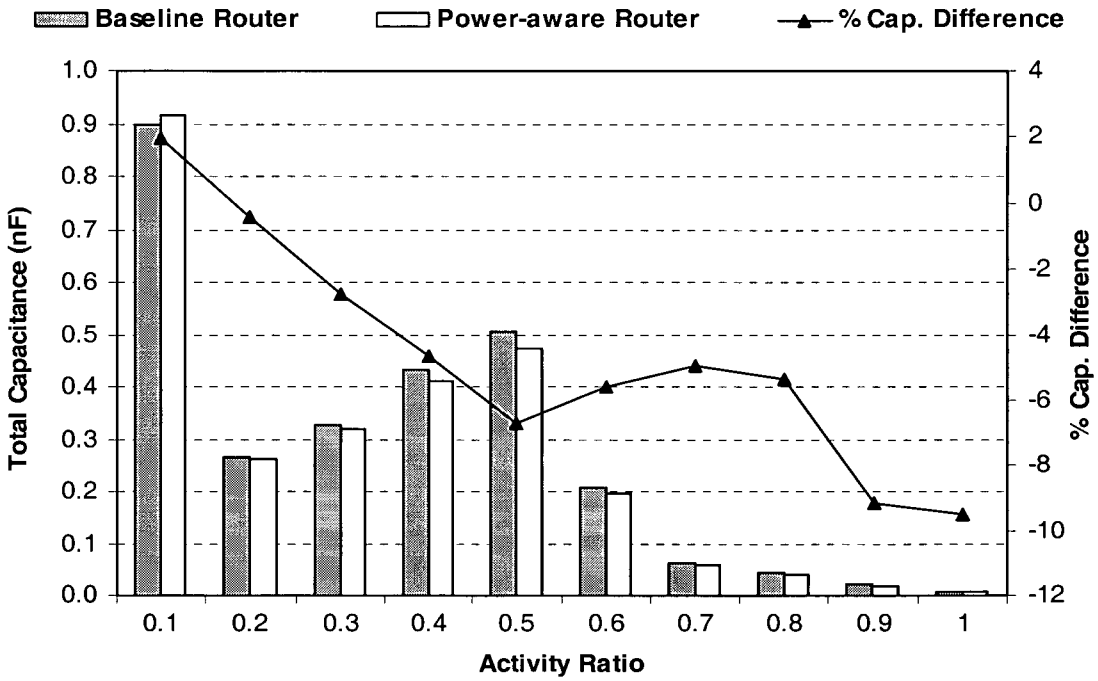


Figure 7.4: P-T-VRoute results (wire cap. vs. switching activity).

7.5 Summary

This chapter described how energy dissipation can be minimized during the routing stage of the FPGA CAD flow. It then described a new power-aware router that is based on the original VPR router. The power-aware routing algorithm minimizes energy by minimizing the amount of routing circuitry used to route high activity nets. Finally, the energy dissipation of circuits routed using the power-aware router was compared to that of circuits routed using the original VPR router. The results were similar to those achieved by the power-aware placement algorithm. The average energy dissipation was reduced by 2.6%, where all of the 20 benchmarks showed an improvement. The following chapter examines the energy reductions when the power-aware algorithms are applied within the same CAD flow.

Chapter 8

COMBINED RESULTS

The four previous chapters considered each FPGA CAD algorithm in isolation in order to determine how suitable each algorithm is to energy minimization. This chapter combines the power-aware algorithms described in the previous chapters in order to examine the interactions between the reductions of each power-aware algorithm.

8.1 Discussion

Intuitively, we would not expect the energy reductions of the individual power-aware algorithms to be perfectly cumulative when the algorithms are combined. All the power-aware algorithms described in the previous chapters minimize energy using similar techniques. Generally, each algorithm reduces the capacitance of the high activity nets or the total number of nets. When these algorithms are applied successively, the later algorithms are likely to be less effective at minimizing energy since they have fewer high activity nets to minimize.

8.2 Experimental Methodology

To compare the influence of the power-aware clustering algorithm on the energy dissipation, we use the experimental framework described in chapter 3; however, instead of replacing only one algorithm at a time, we replace two or more algorithms with their power-aware counterparts. For each combination of power-aware and baseline algorithm, however, each benchmark circuit

is still passed through the entire FPGA CAD flow before estimating the power and critical-path delay of each circuit.

8.3 Experimental Results

The results for all twelve possible CAD algorithm combinations are presented in Table 8.1.

Table 8.1: Combined Results (Energy nJ).

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

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, we determine the overlap for all remaining power-aware pairings. The results are shown in Table 8.2.

Table 8.2: 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 8.2 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 8.1, where each point corresponds to one benchmark circuit and the line is a linear regression trend line.

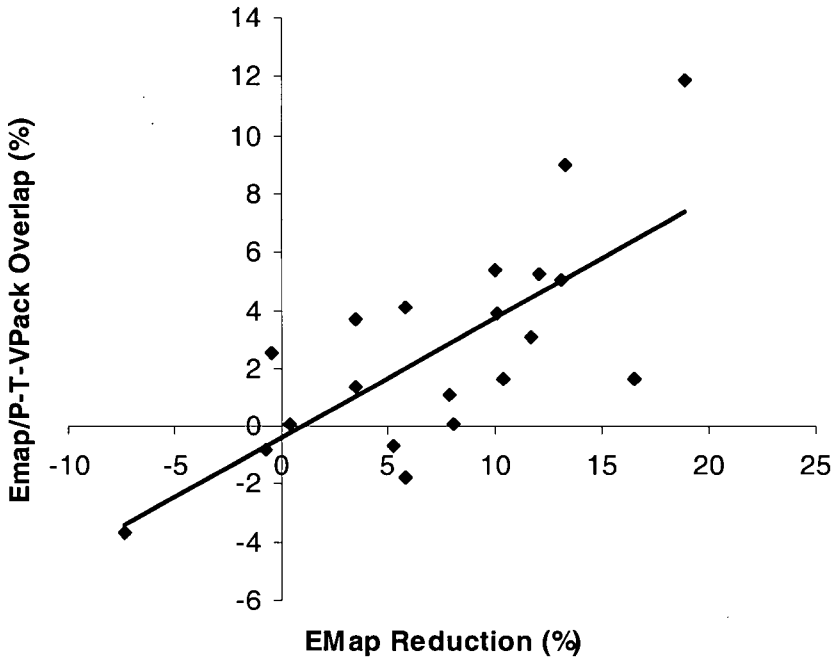


Figure 8.1: EMap/P-T-VPack Overlap.

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

8.4 Summary

This chapter combined the power-aware algorithms describes in the previous chapter in order to investigate the interactions between the reductions achieved by each CAD algorithm. The results indicate that the reductions of the entire power-aware CAD flow are mostly cumulative, with only 3.2% overlap overall. Most of the overlap originates from the interaction of the technology mapping and clustering stages, since the two algorithms account for most of the

overall reductions. Finally, is was shown that the overlap between stages increases proportionally with respect to the reductions of the previous stages.

Chapter 9

CONCLUSIONS AND FUTURE WORK

9.1 Summary and Contributions

In this thesis, we have investigated the interactions between various power-aware FPGA CAD algorithms. The energy reduction of the individual and combined algorithms was measured using very detailed models, which estimate the power dissipation and critical-path delay of circuits after they have been placed and routed onto FPGAs. The individual energy reductions 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 energy reduction was achieved during the technology mapping and clustering stages of the power-aware FPGA CAD flow. Furthermore, we observed that the energy reductions achieved during the earlier stages (technology mapping and clustering) originated primarily from minimizing the amount of connections between logic blocks, whereas the energy reductions achieved during the later stages (placement and routing) originated primarily from minimizing the capacitance of high-activity connections.

After measuring the energy reduction of each power-aware algorithm in isolation, the algorithms were combined to determine if the energy reductions were cumulative. The results indicate that the energy reductions, when the power-aware algorithms are combined, are mostly cumulative, with only 3.2% overlap overall. Furthermore, we observed that most of the overlap originates

from the interaction between the technology mapping and clustering stages, since these two algorithms account for most of the overall reductions. Of course, the numerical results are specific for our algorithms; however, we expect that other power-aware FPGA CAD flows would produce similar conclusions. We have not yet considered high-level synthesis, but we expect that the reductions achieved there could be significant.

Finally, the 22.6% energy reduction that is achieved when all four power-aware algorithms are employed corresponds to a 32% power reduction and a 14% critical-path delay increase. To achieve the same power reduction by simply slowing the system clock, circuit delay would have to be increased by 47%. Therefore, for designs constrained by a power budget, using a power-aware FPGA CAD flow to reduce power has significantly less impact on performance than slowing the system clock (33% less for this CAD flow).

9.2 Future Work

Although very relevant to industry, this work is only a preliminary step towards a complete power-aware CAD flow for modern FPGAs. In the study, FPGA CAD algorithms were optimized to minimize energy dissipation of FPGAs that consist of programmable logic blocks and programmable routing only. Similarly, the performance of these algorithms was measured using power and delay models that are also based on this academic model. Modern FPGAs being introduced by the leading FPGA companies are deviating increasingly from this model. In addition to programmable logic blocks and programmable routing, new FPGAs have additional features, such as multiple clock domains, embedded memories, embedded arithmetic logic units, and even embedded processors, as shown in Figure 9.1. These new features introduce new power consumption issues, many of which can only be tackled at the system level. Moreover, as

process technology continues to scale, static power consumption caused by transistor leakage current is becoming increasingly important [65]. Thus, power should be optimized in two ways: by examining how higher-level FPGA CAD tools can optimize for power at the system level, and by examining how CAD tools can reduce the amount of static power dissipated by an FPGA implementation.

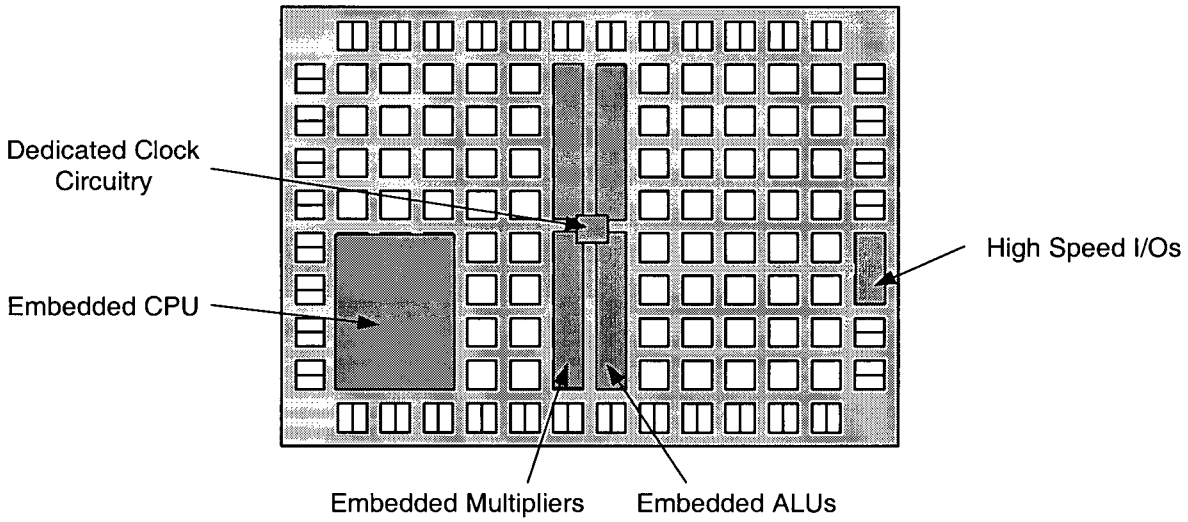


Figure 9.1: Modern FPGA with Embedded System-Level Blocks.

9.2.1 System-Level Power Optimization for FPGAs

Although there is room for power optimization at the physical design level, significant energy reductions can only be achieved by optimizing the design at the system level. Modern FPGAs contain processors, memories, DSP blocks, and other system-level components. High-level design tools must partition systems between these tasks. Unlike existing system-level power optimization tools, which can select high-level blocks from a library, these blocks are pre-fabricated in an FPGA. The task is therefore to use existing blocks in the most energy-efficient way. In many cases, the blocks are configurable. Memories, for example, can usually be used in

one of many modes; the mode used will significantly impact the way data structures can be stored in the memory. As another example, processors in FPGAs are flexible; finding the right balance between these flexible (but pre-fabricated) processors and FPGA logic has not received significant attention.

A related feature found in modern FPGAs is the complex clock distribution system. Modern FPGAs typically contain support for many clock domains. In [16], it was shown that the clock distribution system in an FPGA with only a single clock domain consumes a significant amount of power; it is likely that the amount of power consumed by clock distributions with more clock domains is even more significant. The use of these domains to optimize for power is another system-level issue that has not received attention.

An important part of this future work will be the use of very detailed power models. The existing model we have been using, while very detailed, does not take into account processors, memories, and multiple clock domains. Thus, this model should be extended to accurately estimate the power consumption of these system-level blocks.

9.2.2 Static Power Optimizations

As process technology continues to scale, static power consumption caused by transistor leakage current is becoming increasingly important. FPGA vendors estimate that static power will account for half of the total power dissipated by the next generation of FPGAs.

Circuit level techniques, such as increasing transistor length and increasing threshold voltage, can be used to reduce transistor leakage current; however, these techniques typically incur area

and performance penalties. A given FPGA architecture could hypothetically contain some high-speed programmable circuitry and some low-power programmable circuitry. Power-aware CAD tools could then implement critical-path logic using high-speed circuitry and the remaining logic using low-power circuitry to reduce overall power consumption while maintaining the same performance.

Another technique, which involves turning off parts of a design while they are not required, is promising as well. By disconnecting a sub-circuit from the power supply, static power consumption is eliminated. This technique can be applied to fixed integrated circuits in a relatively straight forward manner. However, this technique is more intricate for circuits implemented using programmable logic. Circuits implemented using FPGAs are partitioned into small logic elements and then placed to physical locations on the FPGA. To enable this power saving scheme, the FPGA would need to be divided into regions that could be turned on or off. During the placement stage of the FPGA CAD flow, logic elements that are likely to be turned on and off at the same time would then need to be placed together, thereby increasing the likelihood that a section can be turned off. As the division size becomes smaller this likelihood increases; however, the overhead incurred by the added circuitry, required to turn the regions on or off, increases.

The feasibility of static power reducing techniques such as these can only be determined by careful analysis at the system-level.

REFERENCES

[1] A.H. Farrahi, and M. Sarrafzadeh, "FPGA technology mapping for power minimization", International Workshop on Field-Programmable Logic and Applications, pp. 167-174, 1994.

[2] M.J. Alexander, "Power Optimization for FPGA Look-Up Tables", ACM International Symposium on Physical Design, pp. 156-162, 1997.

[3] C-C. Wang, and C-P Kwan, "Low Power Technology Mapping by Hiding High-Transition Paths in Invisible Edges for LUT-Based FPGAs", IEEE International Symposium on Circuits and Systems, pp. 1536-1539, June 1997.

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

[5] Z-H. Wang, E-C. Liu, J. Lai, and T-C. Wang, "Power Minimization in LUT-Based FPGA Technology Mapping", ACM Asia South Pacific Design Automation Conference, pp. 635-640, 2001.

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

[7] A. Singh, and M. Malgorzata, "Efficient Circuit Clustering for Area and Power Reduction in FPGAs", Proc. ACM International Symposium on Field-Programmable Gate Arrays, Monterey, CA, pp. 59-66, February 2002.

[8] 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.

[9] N. Togawa, K. Ukai, M. Yanagisawa, and T. Ohtsuki, "A Simultaneous Placement and Global Routing Algorithm for FPGAs with Power Optimization" IEEE Asia Pacific Conference on Circuits and Systems, pp. 125-128, 1998.

[10] B. Kumthekar, and F. Somenzi, "Power and Delay Reduction via Simultaneous Logic and Placement Optimization in FPGAs", Design, Automation, and Test in Europe Conference, pp. 202-207, 2000.

[11] J. Cong, and Y. Hwang, "Simultaneous Depth and Area Minimization in LUT-Based FPGA Mapping", ACM International Symposium on Field-Programmable Gate Arrays, Monterey, CA, pp. 68-74, February 1995.

[12] A. Marquardt, V. Betz, and J. Rose, "Using Cluster-based Logic Blocks and Timing-Driven Packing to Improve FPGA Speed and Density", ACM International Symposium on Field-Programmable Gate Arrays, pp. 37-46, February 1999.

[13] A. Marquardt, V. Betz, and J. Rose, "Timing-Driven Placement for FPGAs", ACM International Symposium on Field-Programmable Gate Arrays, Monterey, CA, pp. 203-213, February 2000.

[14] V. Betz, "Architecture and CAD for the Speed and Area Optimization of FPGAs", Ph.D. Dissertation, University of Toronto, 1998.

[15] V. Betz, J. Rose, and A. Marquardt, "Architecture and CAD for Deep-Submicron FPGAs", Kluwer Academic Publishers, 1999.

[16] K. Poon, A. Yan, and S. Wilton, "A Flexible Power Model for FPGAs", International Conference on Field-Programmable Logic and Applications, September 2002.

[17] J. Greene, E. Hamdy, and S. Beal, "Antifuse Field Programmable Gate Arrays", Proceedings of the IEEE, pp. 1042-1056, July1993.

[18] S. Brown, "An Overview of Technology, Architecture and CAD Tools for Programmable Logic Devices", Custom Integrated Circuits Conference, pp. 69-76, 1994.

[19] J. Rose, R.J. Francis, D. Lewis, and P. Chow, "Architecture of Field-Programmable Gate Arrays: The Effect of Logic Functionality on Area Efficiency", IEEE Journal of Solid-State Circuits, 1990.

[20] E. Ahmed and J. Rose, "The effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density", ACM International Symposium on Field-Programmable Gate Arrays, pp. 3-12, 2001.

[21] E. Lin, "Product Term Mode Embedded Memory Arrays: Architectures and Algorithms", M.A.Sc. Thesis, University of British Columbia, 2001.

[22] G. Lemieux and D. M. Lewis. Analytical framework for switch block design. In International Conference on Field-Programmable Logic (FPL), pages 122-131, 2002.

[23] G. Lemieux, "Efficient Interconnection Network Components for Programmable Logic Devices", Ph.D. Dissertation, University of Toronto, 2003.

[24] L.R. Ford and D.R. Fulkerson, "Flows in Networks", Princeton, NJ: Princeton University Press, 1962.

[25] J. Cong, Y. Ding, "On Area/Depth Trade-off in LUT-Based FPGA Technology Mapping", IEEE Transactions on VLSI Systems, Vol. 2, No. 2, pp. 137-148, June 1994.

[26] J. Cong, and Y. Hwang, "Structural Gate Decomposition for Depth-Optimal Technology in LUT-based FPGA Designs", TODAES , Vol. 5, no. 3, July 2000.

[27] F.N. Najm, "Transition Density, A New Measure of Activity in Digital Circuits", Texas Instruments Technical Report #7529/0032, August 1991.

[28] M. Nemani and F. Najm, "Towards a High-Level Power Estimation Capability", IEEE Transactions on Computer-Aided Design, Vol. 15, No. 6, pp. 588-598, June 1996.

[29] G.K Yeap, "Practical Low Power Digital VLSI Design", Kluwer Academic Publishers, 2001.

[30] D. Lewis, V. Betz, et al., "The Statix™ Routing and Logic Architecture", ACM International Symposium on Field-Programmable Gate Arrays, pp. 12-20, February 2002.

[31] Xilinx, Vertex-II Pro Platform FPGAs: Functional Description, ver. 2.0, June 13, 2002.

[32] J. Cong, L.W. Hagen, and A.B. Kahng, "Random Walks for Circuit Clustering", IEEE Conference on Application Specific Integrated Circuits, pp. 14.2.1-14.2.4, June 1991.

[33] J. Cong and M. Smith, "A Parallel Bottom-up Clustering Algorithm with Applications to Circuits Partitioning in VLSI Design", ACM/IEEE Design Automation Conference, pp.755-60, 1993.

[34] J. Cong and S.K. Lim, "Edge Separability based Circuit Clustering with Application to Circuit Partioning", ACM/IEEE Asia South Pacific Design Automation Conference, pp. 429-434, 2000.

[35] L.W. Hagen and A.B. Kahng, "Combining Problem Reduction and Adaptive Multi-Start: A New Technique for Superior Iterative Partitioning", IEEE Transactions on Computer-Aided Design, pp. 709-717, 1997.

[36] D.J.H Huang and A.B. Kahng, "When Clusters Meet Partitions: New Density-Based Methods for Circuit Decomposition", European Design and Test Conference, pp. 60-64, 1995.

[37] B.S. Landman and R.L. Russo, "On a pin versus block relationship for partitions of Logic Graphs", IEEE Transactions on Computers, C-20, pp. 1469-1479, 1974.

[38] A. Dunlop and B. Kernighan, "A Procedure for Placement of Standard-Cell VLSI Circuits", IEEE Transactions on Computer-Aided Design, pp. 92-98, 1985.

[39] D. Huang and A. Kahng, "Partitioning-Based Standard-Cell Global Placement with an Exact Objective," ACM Symposium on Physical Design, pp. 18-25, 1997.

[40] J. Rose, W. Snelgrove and Z. Vranesic, "ALTOR: An Automatic Standard Cell Layout Program", Canadian Conference on VLSI, pp. 169-173, 1985.

[41] J. Kleinhans, G. Sigl, F. Johannes, and K. Antreich, "Gordian: VLSI Placement by Quadratic Programming and Slicing Optimization", IEEE Transactions on Computer-Aided Design, pp. 356-365, 1991.

[42] A. Srinivasan, K. Chaudhary, and E. Kuh, "Ritual : A Performance Driven Placement Algorithm for Small Cell ICs", International Conference on Computer Aided Design, pp. 48-51, 1991.

[43] B. Riess and G. Ettelt, "Speed: Fast and Efficient Timing Driven Placement", IEEE International Symposium on Circuits and Systems, pp. 377-380, 1995.

[44] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by Simulated Annealing", Science, pp. 671-680, 1983.

[45] C. Sechen and A. Sangiovanni-Vincentelli, "TimberWolf Placement and Routing Package", JSSC, pp. 510-522, 1985.

[46] M. Huang, F. Romeo, and A. Sangiovanni-Vincentelli, "An Efficient General Cooling Schedule for Simulated Annealing", International Conference on Computer Aided Design, pp. 381-384, 1986.

[47] J.S. Rose, "Parallel Golbal Routing for Standard Cells", IEEE Transactions on Computer Aided Design, pp.1085-1095, 1990.

[48] Y. Chang, S. Thakur, K. Zhu, and D. Wong, "A New Global Routing Algorithm for FPGAs", International Conference on Computer Aided Design, pp. 356-361, 1994.

[49] S. Brown, J. Rose, Z.G. Vranesic, "A Detailed Router for Field-Programmable Gate Arrays", IEEE Transactions on Computer Aided Design, pp. 620-628, 1992.

[50] G. Lemieux, S. Brown, "A Detailed Router for Allocating Wire Segments in FPGAs", ACM Physical Design Workshop, pp. 215-226, 1993.

[51] G. Lemieux, S. Brown, D. Vranesic, "On Two-Step Routing for FPGAs", ACM Symposium on Physical Design, pp. 60-66, 1997.

[52] M. Placzewski, "Plane Parallel A* Maze Router and Its Application to FPGAs", ACM Design Automation Conference, pp. 691-697, 1990.

[53] L. McMurchie, and C. Ebeling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs", ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, pp. 111-117, February 1995.

[54] Y.-L. Wu, M. Marek-Sadowska, "An Efficient Router for 2-D Field-Programmable Gate Arrays", European Design Automation Conference, pp. 412-416, 1994.

[55] Y.-S. Lee, A. Wu, "A Performance and Routability Driven Router for FPGAs Considering Path Delays", ACM Design Automation Conference, pp. 557-561, 1995.

[56] L. Shang, A.S. Kaviani, K. Bathala, "Dynamic Power Consumption in Vertex-II FPGA Family", Tenth ACM International Symposium on Field-Programmable Gate Arrays, pp. 157-164, February 2002.

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

[58] T. Okamoto and J. Cong, "Buffered Steinet Tree Construction with Wire Sizing for Interconnect Layout Optimization", International Conference on Computer Aided Design, pp. 44-49, 1996.

[59] F.N. Najm, "Low-pass Filter for Computing the Transition Density in Digital Circuits", IEEE Transactions on Computer-Aided Design, vol. 13, no. 9, pp. 1123-1131, September 1994.

[60] S.J.E Wilton, N.P. Jouppi, "CACTI: An Enhanced Cache Access and Cycle Time Model", IEEE Journal of Solid-State Circuits, vol. 31, no. 5, pp. 677-687, May 1996.

[61] Altera, Stratix Programmable Logic Device Family Data Sheet, ver. 2.0, April 2002.

[62] J.M. Rabaey, "Digital Integrated Circuits: A Design Perspective", Prentice-Hall, 1996.

[63] K.Y. Toh, P.K. Ko, and R.G. Meyer, "An Engineering Model for Short-Channel MOS Devices", IEEE Journal of Solid-State Circuits, vol. 23, no. 4, August 1988.

[64] J. Cong, C. Wu, and E. Ding, "Cut Ranking and Pruning: Enabling A General And Efficient FPGA Mapping Solution", ACM International Symposium on Field-Programmable Gate Arrays, pp. 29-35, February 1999.

[65] T. Tuan and B. Lai, "Leakage Power Analysis in a 90nm FPGA", to appear in the IEEE Custom Integrated Circuits Conference, 2003.

[66] J. Rose and S. Brown, "Flexibility of interconnection structures for field-programmable gate array", ISSC, Vol. 26, pp. 277-282, Mar. 1991.

[67] J. Cong and Y. Ding, "FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs", IEEE Trans. on Computer-Aided Design, vol. 13, no. 1, pp. 1-12, January 1994.