Low-Power Technology Mapping for FPGA Architectures with Dual Supply Voltages

Deming Chen, Jason Cong

Computer Science Department

University of California, Los Angeles

{demingc, cong}@cs.ucla.edu

Fei Li, Lei He

Electrical Engineering Department

University of California, Los Angeles

{feil, lhe}@ee.ucla.edu

ABSTRACT

In this paper we study the technology mapping problem of FPGA architectures with dual supply voltages (Vdds) for power optimization. This is done with the guarantee that the mapping depth of the circuit will not increase compared to the circuit with a single Vdd. We first design a single-Vdd mapping algorithm that achieves better power results than the latest published low-power mapping algorithms. We then show that our dual-Vdd mapping algorithm can further improve power savings by up to 11.6% over the single-Vdd mapper. In addition, we investigate the best low-Vdd/high-Vdd ratio for the largest power reduction among several dual-Vdd combinations. To our knowledge, this is the first work on dual-Vdd mapping for FPGA architectures.

Categories and Subject Descriptors

B.6.3 [Logic Design]: Design Aids – Optimization

General Terms

Algorithms, Design, Performance

Keywords

Technology mapping, low-power FPGA, dual supply voltage

1. INTRODUCTION

Power consumption has become a limiting factor in both high performance and mobile applications. Independent of application, desired performance is achieved by maximizing operating frequency under power constraints. These constraints may be dictated by battery life, chip packaging and/or cooling costs. It is important to minimize power consumption of FPGA chips particularly, because FPGA chips are power inefficient compared to logically equivalent ASCI chips. The main reason is that FPGAs use a large number of transistors to provide programmability. The large power consumption of FPGAs prevents FPGA designs from entering many low-power

applications. Since multimillion-gate FPGAs have become a reality, reducing power consumption at every design and synthesis level is a mandate so that the power dissipation of FPGA chips can be restrained.

One of the popular design techniques for power reduction is to lower supply voltage, which results in a quadratic reduction of power dissipation. However, the major drawback is the negative impact on chip performance. A multiple supply voltage design in which a reduction in supply voltage is applied only to non-critical paths can save power without sacrificing performance. Clustered voltage scaling (CVS) was first introduced in [1], where clusters of high-Vdd cells and low-Vdd cells were formed, and the overall performance was maintained. The work in [2] used a maximum-weighted independent set formulation combined with CVS and gate sizing to enhance power savings on the whole circuit. In [3], a dual supply voltage scaling (DSVS) methodology was designed. The work in [4] introduced variable supply-voltage combined with CVS. It also derived a rule for optimal low Vdd given a high Vdd. It showed that the low Vdd could always be set at a 0.6-0.7 range of the high Vdd to minimize power. The work in [5] assigned variable voltages to functional units at the behavioral synthesis stage. The goal was to minimize the system's power and meet the total timing constraint.

In this work we will study the power minimization problem at the logic synthesis level. Specifically, we will work on technology mapping for FPGA circuits using dual supply voltages. For LUT (*lookup table*)-based FPGAs, technology mapping converts a given Boolean circuit into a functionally equivalent network comprised only of LUTs. The technology mapping for power minimization has been shown to be NP-complete [6] to solve.

There are previous works on technology mapping for low-power FPGA designs, all assuming single Vdd [7,8,9,10]. The basic approach was to hide the nodes of high-switching activity into LUTs so the overall dynamic power was reduced.

In our work we develop a low-power FPGA mapping algorithm, named *DVmap*, with consideration of delay and power optimization crossing two supply voltages. The voltages are denoted as V_L for low Vdd and V_H for high Vdd. We do not add the constraint that the V_L and the V_H nodes have to be clustered separately since FPGA architecture can program the voltages of the build-in LUTs and converters as needed. We use the cut-enumeration technique to produce all the possible ways of mapping a LUT rooted on a node. We then generate different sets of power and delay solutions for each possible way based on the various voltage changing scenarios. After the timing constraint is

determined, the non-critical paths will be relaxed in order to accommodate V_L LUTs to reduce power while maintaining the timing constraint. To show the efficiency of our algorithm, we first design a mapping algorithm with single Vdd, which uses similar cost function as that in DVmap and relaxes the non-critical paths based on cost to achieve better power results. The single-Vdd mapper, named *SVmap*, shows an advantage over the latest published low-power mapping algorithm *Emap* [10] and another low power mapper presented in [8]. We then show that our dual-Vdd mapping algorithm DVmap can further improve SVmap by up to 11.6% for power savings.

The rest of this paper is organized as follows. In Section 2, we provide some basic definitions and formulate the dual-Vdd FPGA mapping problem. Section 3 introduces our FPGA architecture and power model. Section 4 gives the detailed description of our algorithm. Section 5 presents experimental results, and Section 6 concludes this paper.

2. DEFINITIONS AND PROBLEM FORMULATION

A Boolean network can be represented by a directed acyclic graph (DAG) where each node represents a logic gate, and a directed edge *(i, j)* exists if the output of gate *i* is an input of gate *j*. A *PI* (primary input) node has no incoming edges and a *PO* (primary output) node has no outgoing edges. We use *input(v)* to denote the set of nodes which are *fanins* of gate *v*. Given a Boolean network *N*, we use C_v to denote a *cone* of node *v* in *N*. C_v is a sub-network of *N* consisting of *v* and some of its predecessors such that for any node $w \in C_v$, there is a path from *w* to *v* that lies entirely in C_v. The maximum cone of *v*, consisting of all the *PI* predecessors of *v*, is called a *fanin cone* of *v*, denoted as F_v. We use *input(C_v)* to denote the set of distinct nodes outside C_v which supply inputs to the gates in C_v. A *cut* is a partitioning *(X, X')* of a cone C_v such that *X'* is a cone of *v*. *v* is the *root* node of the cut. The node *cut-set* of the cut, denoted *V(X, X')*, consists of the inputs of cone *X'*, or *input(X')*. A cut is *K-feasible* if *X'* is a *K*-feasible cone. In other words, the cardinality of the cut-set (or *cut size* of the cut) is $\leq K$. The *level* of a node *v* is the length of the longest path from any *PI* node to *v*. The level of a *PI* node is zero. The *depth* of a network is the largest node level in the network. A Boolean network is *K-bounded* if $|input(v)| \leq K$ for each node *v*.

Because the exact layout information is not available during the technology mapping stage, we model each interconnection edge in the Boolean network as having a constant delay. Therefore, we approximate the largest delay of the mapped circuit with a *unit delay* model, where each LUT on the critical path (the path with the longest delay) contributes a one-unit delay (single-Vdd case). This largest optimal delay of the mapped circuit is also called the *mapping depth* of the circuit.
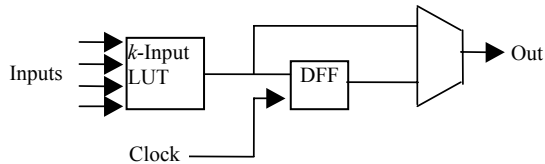
The dual-Vdd mapping problem for min-power FPGA (**DVMF**

Figure 1: Basic logic element (BLE)

Configuration	Worst-Case Delay (ns)	Energy per Switch (J)	Static Power (w)
Vdd 1.3v	0.195	6.36E-14	4.25E-06
Vdd 1.0v	0.240	4.54E-14	4.70E-06
Vdd 0.9v	0.276	3.94E-14	4.50E-06
Vdd 0.8v	0.304	3.70E-14	4.81E-06

Table 1: Delay and power data for logic cell (4-LUT)

problem) is to cover a given *K*-bounded Boolean network with *K*-feasible cones or equivalently, *K*-LUTs, in such a way that the total power consumption is minimized under a dual supply voltage FPGA architecture model, while the optimal mapping depth is maintained.

We assume that the input networks are all 2-bounded and *K* is 4 in this study. Therefore, our final mapping solution is a DAG in which each node is a 4-feasible cone (4-LUT) and the edge *(C_u, C_v)* exists if *u* is in *input(C_v)*. We pick 4-LUT because it is the most commonly used among commercial FPGAs [11] [12]. Our algorithm will work for any reasonable *K* values.

3. ARCHITECTURE AND POWER MODEL

3.1 Logic Element and Level Converter

Figure 1 shows the simplified model of the basic logic cell of a *K*-LUT-based FPGA. The output of the *K*-LUT can be either registered or unregistered. We obtain the delay and power data of a 4-LUT for various supply voltages through SPICE simulation under 0.1um technology. Table 1 shows details. The *worst-case delay* shows the largest time difference between the point that a signal arriving at one of the inputs of the LUT and the point that the LUT generates an output. *Energy_per_switch* represents the energy a whole LUT consumes as a unit per switch of the LUT output (the output is properly buffered). Static power shows the power consumption of the whole LUT if there is no switching in the cycle.
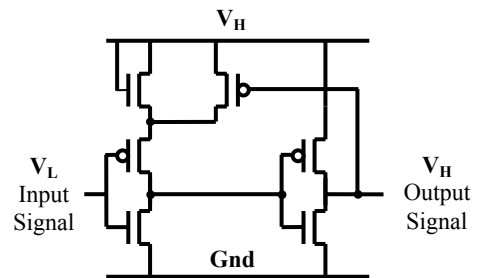

Figure 2: Schematic of a level converter with single supply voltage

A level converter is required when a V_L device output is to be connected to a V_H device input. Otherwise, excessive leakage power will occur in a direct connection. We use the level converter with single supply voltage as proposed in [13]. We show the transistor level schematic in Figure 2. A V_L input signal is converted into a V_H output signal while the level converter only uses a single supply voltage V_H. Table 2 shows the detailed power

Configuration	Worst-Case Delay (ns)	Energy per Switch (J)	Static Power (w)
1.0v to 1.3v	0.0814	7.40E-15	1.04E-07
0.9v to 1.3v	0.0801	8.05E-15	1.39E-07
0.8v to 1.3v	0.0845	9.73E-15	2.40E-07

Table 2: Delay and power data for the level converter

and delay data for the converter. Notice that the delay of 0.9v/1.3v is smaller than that of 1.0v/1.3v. This is because we size the transistors in the level converter differently for different V_L/V_H combinations to achieve better delay and power. Therefore, the delay and power trends cannot be simply predicted.

It has been shown that cluster-based logic blocks can improve the FPGA performance, area and power [14,15]. We insert level converters into the configurable logic block (CLB). Figure 3 shows such a CLB containing N BLEs. The output of an LUT can be programmed to go through a level converter or bypass it. This gives us the capability to insert a level converter between a V_L BLE and a V_H BLE, regardless of the two BLEs being in the same cluster or not. SPICE simulation shows that the power consumption of the MUX associated with the converter is about one fifth of that of a converter. The delay of the MUX is 0.014ns, which is almost ignorable. We assume that there are pre-fabricated tracks in the routing channels with either V_H or V_L settings. When a V_L BLE is driving the routing interconnects (wires), we assume that it uses the V_L routing tracks, i.e., the supply voltage for interconnects and the connecting buffers is V_L.

3.2 Power Model
Both dynamic and static power is considered for LUTs, level converters, and wires and buffers in the routing tracks. For each K-feasible cone (a K-cut), the total power of the cone is calculated as follows:

$$P_{cone} = S_o \cdot P_{LUT} + (1 - S_o)P_{LUT\_static} + P_{inputs} + P_{net} \quad (1)$$

where S_o is the switching activity of the cone output. P_{LUT} is *energy_per_switch * f* (circuit frequency). P_{LUT} considers both dynamic and static power. P_{LUT_static} is the static power of an LUT, which is counted when the LUT is not switching. P_{inputs} is the power consumed on the cut inputs, which is defined as follows:

$$P_{inputs} = 0.5f \cdot V_{dd}^2 \sum_i^k S_i \cdot C_{in} \quad (2)$$

where S_i is the switching activity on input i of the cut. C_{in} is the input capacitance on an LUT (a constant). P_{net} is calculated as follows:

$$P_{net} = 0.5f \cdot V_{dd}^2 \cdot C_{net} \cdot S_o + P_{buf\_static} \quad (3)$$

where C_{net} is the estimated output capacitance of wires and buffers contained in the net driven by the LUT, and P_{buf_static} is the static power of the buffers contained in the net. C_{net} is changeable gate by gate. To obtain reasonable wire-capacitance estimation before placement and routing, we profile a series of benchmarks using *VPR* [14] as the placement and routing tool. Figure 4 shows the profiling data with the 20 largest MCNC benchmarks as used

by the VPR package. There is an obvious correlation between the *fanout* number of the gates and the wire length of the net driven by the gates after placement and routing. The wire length is in the unit of wire segment, each of which is across one CLB of size 4. There are buffers between the wire segments. Figure 5 shows the average wire length across 20 benchmarks for each fanout number when the fanout number is ≤ 20. The correlation can be considered as linear in Figure 5. Since most of the gates have relatively small fanout numbers, we will use the plotted trend line in Figure 5 to estimate the net capacitance on the gate output.[1]
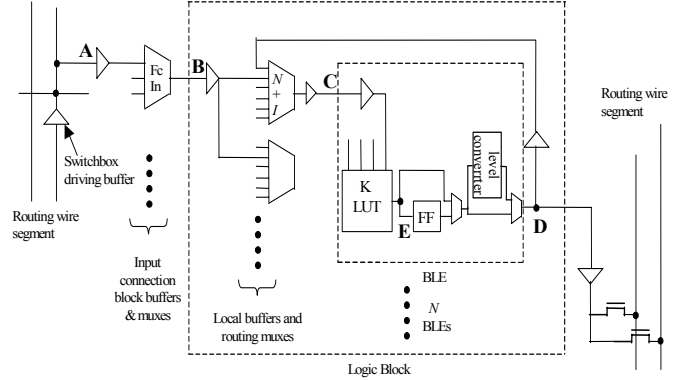

Figure 3: A CLB with inserted level converters

Because we place the level converter on the LUT output, we need to handle a special case when a *driver* LUT is driving multiple fanout LUTs (*end* LUTs) of mixed voltage settings. If the driver LUT is using V_L, the converter for the driver LUT will convert V_L to V_H before the driving signal goes into the routing channel. If the driver LUT is using V_H, there is no need to do the conversion. Then, some of the driver LUT's fanouts that connect to V_L end LUTs should go through V_L routing tracks, and the fanouts that drive V_H end LUTs should go through V_H routing tracks. We count the V_H fanouts and V_L fanouts of the driver LUT and estimate the P_{net} power separately. The output buffer (on point D in Figure 3) is assigned a voltage of V_H, which works fine because the V_H device output can drive both V_H and V_L device inputs without a problem. In a general case, if a V_L or V_H driver LUT is driving end LUTs that are all using V_L, then there is no need to go through the converter, and the routing tracks between the driver LUT and end LUTs are all using V_L. If the driver LUT is a V_H LUT, and all of the end LUTs are using V_H as well, then all the routing tracks in between are using V_H.

Both S_i and S_o are calculated up front before the mapping starts. We use the switching activity calculator available in SIS [16], which builds BDD (binary decision diagram) for each node in the network, counts the probability of going down each path in the BDD, and sums it up to give the total probability of function being logic value 1. The switching activity for the output of the

---

[1] The actual used capacitance for the net is obtained by multiplying an empirical constant that compensates the difference between the estimated power from the estimated net capacitance and the power calculated based on the RC model after placement and routing.

node v is then calculated by a formula as $2 \cdot Pv \cdot (1 - Pv)$ [17], where Pv is the probability of node v being 1.
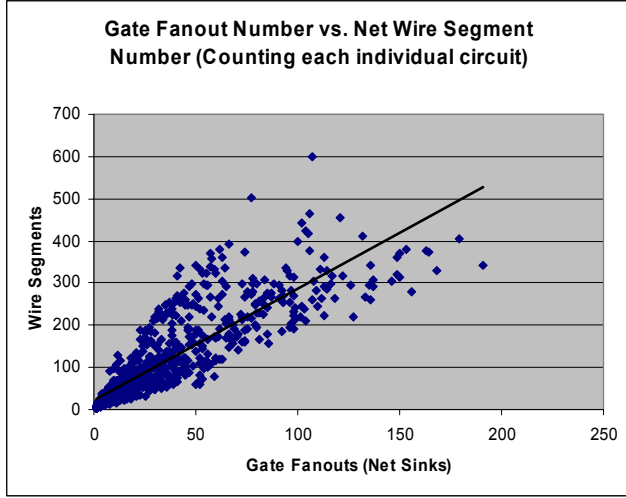

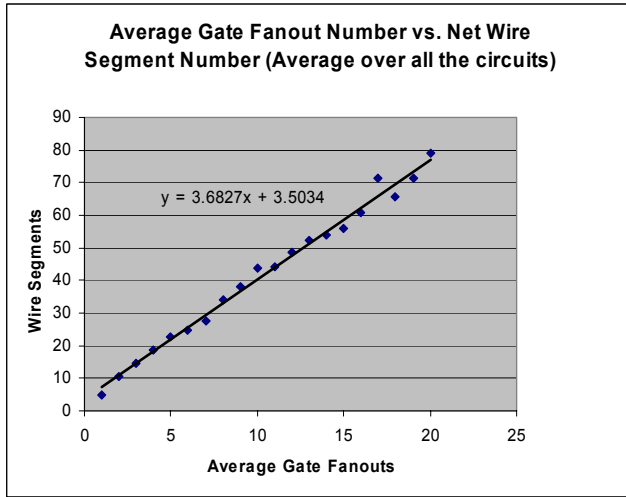Figure 4: Plot of gate fanout number and wire length driven by the gate

Figure 5: Average gate fanout number and wire length correlation for smaller fanout number

4. ALGORITHM DESCRIPTION

4.1 Overview

Cut-enumeration is an effective method to find out all the possible ways of the K-feasible cones rooted on a node. Both [18] and [19] used this method for mapping to minimize area. Works in [8] and [10] are low-power mappers based on this technique as well. A cut can be represented using a product term (or a *p-term*) of the variables associated with the nodes in the node cut-set of $V(X_v, X_v')$. A set of cuts can be represented by a sum-of-product expression using the corresponding p-terms. Cut-enumeration is guided by the following theorem [19]:

$$f(K, v) = \otimes_{u \in input\,(v)}^{k} [u + f(K, u)] \qquad (4)$$

where $f(K, v)$ represents all the K-feasible cuts rooted at node v, operator + is *Boolean OR*, and $\otimes^{K}$ is *Boolean AND* but filtering out all the resulting *p-terms* with more than K variables.

More specifically, every cut rooted on a node can be generated by combining the cuts on the root node's direct fanin nodes. We call the cuts on the fanin nodes *subcuts*. The cut enumeration process will combine one subcut from every fanin node to form a new cut for the root node. If the number of the inputs of the new cut exceeds K, the cut is discarded. For single-Vdd mapping, each cut represents one unit delay. The arrival time for each node is propagated from the *PI* through the consecutive cuts in the fanin cone of the node. We obtain the minimum arrival time for a node v through the arrival times of the cuts rooted on v:

$$Arr_v = \underset{\forall c \text{ on } v}{\text{MIN}} \quad \underset{i \,\in\, input(c)}{[\text{MAX} (Arr_i) + 1]} \qquad (5)$$

where c is a cut generated for v through cut-enumeration. We call the cut, whose arrival time is the smallest among all the cuts, MC_v. Thus, MC_v provides the delay of Arr_v. The minimum arrival time of each node is iteratively calculated until all the *POs* are reached. The longest minimum arrival time of the *POs* is the minimum arrival time of the circuit.

Similarly, we can propagate power through the cut-enumeration process. We can obtain the power associated with a cut c as follows:

$$P_c \quad = \underset{i \,=\, input(c)}{\Sigma [\, P_i / f_i\,]} + U_c \qquad (6)$$

where U_c is the power contributed by cut c itself (to be covered next). f_i is the fanout number of signal i. Therefore, the power on i (the propagated power for F_i) is shared and distributed into other fanout nodes of i. Once the outputs reconverge, the total power of the shared fanin cones will be summed up [19]. This idea tries to estimate the power more accurately, considering the effects of gate fanout. Otherwise, the power of F_i may be counted multiple times while processing the different fanouts of node i. However, since we do not know whether there will be duplications for node i at this point, this model is still a heuristic.[2]

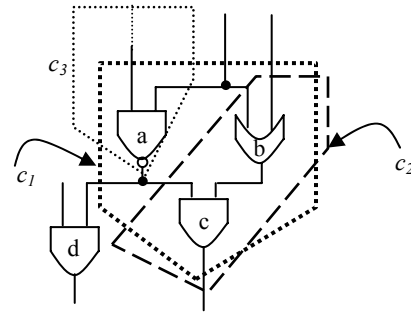

Figure 6: Illustration of various cuts

---

[2] Suppose i has two fanouts and i also gets duplicated in the final mapping result, then the actual power will most likely be larger than what this model estimates.

For single-Vdd architecture, the power for a node v, P_v, is equal to P_c, where $c = MC_v$. Therefore, the powers of the cuts and nodes are iteratively calculated until the enumeration process reaches all the *POs*. We will present the power propagation scenarios for dual-Vdd architectures later. In the next subsection, power calculation for a cut itself, U_c, is introduced first. More precisely, we will call it cost calculation because we are not using the actual power for calculating U_c. We consider other characteristics of the cut to help reduce node duplications and the total number of edges of the mapped circuit. As a result, the power of the cut is minimized when U_c is minimized.

4.2 Calculation of Cut Cost

Although each cut represents one LUT, using a fixed unit cost for a cut will not accurately reflect the property of the cut. Two cuts that have the same cut size may have different characteristics that make the cost of these two cuts different. The characteristics of the cut we consider include node coverage, node duplication, cut size, switching activities and output fanout number. All of these factors influence the cost of the cut.

- *Node Coverage and Duplications*

In Figure 6, cut c_1 and c_2 all have three inputs. However, cut c_1 covers three nodes, and cut c_2 only covers two. Intuitively, c_1 is more preferred because it implements more logic. In other words, the cost of c_1 should be conversely proportional with its node coverage number. On the other hand, c_1 contains the node a, which has two fanouts. This indicates that the cut rooted on node d has to cover node a again, i.e., node a is duplicated once. Duplication generally hurts power minimization [8]. Therefore, this will increase the cost of c_1. We will consider both node coverage and node duplication for a cut to evaluate its cost.

- *Cut Size*

The total number of edges of the mapped circuit plays an important role for the power consumption of the circuit. The larger the number of edges, the more interconnects it produces during placement and routing. Since a large portion of the total circuit power comes from interconnects for FPGAs [15], reducing the total number of edges is an important task during mapping. We try to control the total connections in the cost function. If all the other factors between two cuts are the same, the cut with the larger cut size will have larger cost.

- *Switching Activity*

We accumulate all the switching activity values on the input nodes of a cut and use this sum to penalize cuts that incur large switching power. The smaller this sum, the larger the chance that the cut will be picked. This naturally selects cuts that hide highly switching nodes in LUTs to reduce power. This factor helps to reduce the total connections of the mapping as well, because total switching activity on the inputs is usually proportional to cut size.

- *Output Fanout Number*

The last factor we consider is the fanout number of the root node of the target cut. This is trying to control node duplication from another angle. In Figure 6, cut c_3 should have smaller cost because, unlike cut c_1, it does not generate node duplication for node a. The larger the fanout number, the better for picking this cut, because it potentially saves more duplications.

Based on the factors mentioned above, we design our cost function as follows:

$$Cost_c = \frac{CS_c \bullet (1 + \partial \bullet \sum S_i) \bullet (1 + \partial \bullet DUP_c)}{(1 + COV_c + \partial \bullet FT_c)} \qquad (7)$$

CS_c is the cut size of the cut c. S_i is the switching activity on input i of the cut. DUP_c is the number of potential duplications of c. COV_c is the total number of nodes covered by the cut, and FT_c is the fanout number of the root node. ∂ is a constant.

We use this cost function during the cut-enumeration process. After mapping, the actual power of each mapped LUT is estimated based on the power model presented in Section 3.2.

4.3 Delay and Cost Propagation for Dual-Vdd Consideration

There are four cases between two connected LUTs under dual-Vdd settings. Table 3 shows these cases when LUT_1 is driving LUT_2.

Cases	LUT_1 Vdd	LUT_2 Vdd	Converter
1	V_L	V_L	No
2	V_L	V_H	Yes
3	V_H	V_L	No
4	V_H	V_H	No

Table 3: Dual-Vdd scenarios

During cut enumeration, beside the delay and cost value calculated for the single-Vdd situation, each cut (represented by LUT_2) will have additional power and delay values corresponding to the four cases listed in Table 3. We can name the delay and cost propagation for single Vdd as *case 0* since it gives a baseline solution that provides the optimal mapping depth of the circuit. The dual-Vdd cases will maintain this mapping depth and relax the non-critical path to accommodate V_L LUTs.

For each of the four cases, the delay propagation becomes:

$$Arr_v = \underset{\forall c \text{ on } v}{\text{MIN}} \quad \underset{i \,\in\, input(c)}{[\text{MAX } (Arr_i) + D_{LUT} + \{D_{conv}\}]} \qquad (8)$$

where $[\text{MAX } (Arr_i) + D_{LUT} + \{D_{conv}\}]$ is the arrival time for a cut c rooted on v (v is the root node of LUT_2). Let us examine *case 2* as an example. Arr_i is the arrival time on input i of cut c corresponding to LUT_1's voltage setting, V_L. D_{LUT} is 1 in this case because LUT_2 is using V_H.[3] There will be a converter required between LUT_1 and LUT_2, which contributes a delay of D_{conv}. In the formula, D_{conv} is in braces {} to indicate that it is required only as needed. Arrival time of each cut for *case 2* is calculated first with voltage setting V_H. Then, Arr_v for *case 2* is calculated, and its voltage setting is V_H. We observe that there are two choices for Arr_i generated before from *case 1* and *case 3*, because these two cases provided Arr_i values with V_L setting in the previous delay propagation. We will pick the case that gives smaller MAX (Arr_i) value, and its cost is used for cost

---

[3] D_{LUT} is larger than 1 for LUTs using V_L, proportional to the SPICE data shown in Section 3.1.

propagation. If these two cases provide the same delay, the case with smaller cost will be picked for cost propagation.[4]

Cost propagation for each case for the cut is as follows:

$$P_{c\text{-}Vdd} = \Sigma \ [\ P_i \ / f_i] + U_{c\text{-}Vdd} + \{U_{conv}\} \qquad (9)$$
$$i = input(c)$$

P_i is the propagated cost on input i with LUT_1's voltage setting. $U_{c\text{-}Vdd}$ is the cost of c (LUT_2) itself. It can be either $U_{c\text{-}VL}$ or $U_{c\text{-}VH}$, depending on LUT_2's voltage setting. The value of $U_{c\text{-}VL}$ is the same as the one defined for single Vdd, U_c. $U_{c\text{-}VH}$ is proportionally larger than $U_{c\text{-}VL}$ as follows:

$$U_{c\text{-}VH} = (Power_{c\text{-}VH} \ / \ Power_{c\text{-}VL}) \bullet U_{c\text{-}VL} \qquad (10)$$

where $Power_{c\text{-}VH}$ and $Power_{c\text{-}VL}$ are actual power consumption values for cut c when assigned with V_H and V_L hypothetically. They are calculated through the power estimation model in Section 3.2. This gives an accurate proportional increase of $U_{c\text{-}VH}$ over $U_{c\text{-}VL}$. U_{conv} is counting both dynamic and static power of the level converter when it is needed. When it is not needed, only static power is counted. The dynamic and static power of the MUX associated with the converter is always counted.

In addition, we have a voltage setting for each of the four cases, $V_c = V_{LUT2}$. The delay, cost and voltage calculation propagates from *PIs* to *POs* iteratively. The Arr_v and $P_{c\text{-}Vdd}$ will become Arr_i and P_i for next iteration during the calculation.

4.4 Mapping Generation

After cut enumeration, a mapping procedure is carried out guided by the required time, which is the optimal mapping depth of the network. The critical path is always driven by V_H, and only non-critical paths can be driven by V_L to reduce power under the condition that they will not violate the required time of the network. First, all the primary outputs are mapped, then the inputs of the generated LUTs are mapped.

Before the mapping starts, we set nodes with large fanout numbers as *tentative* LUT roots, i.e., the cuts rooted on these nodes have a much higher chance of being selected in the
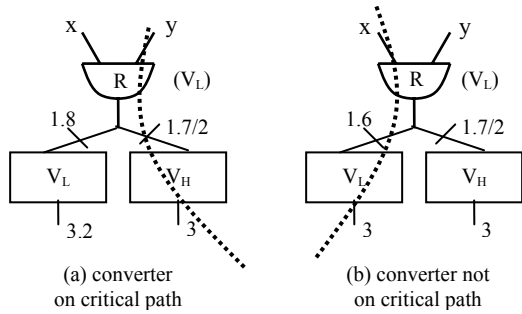

(a) converter
on critical path

(b) converter not
on critical path

Figure 7: Critical path and level converter delay.
Numbers are required times.

---

[4] Here, we only show *case 2* as an example. Other cases are similarly handled. Each individual case will have its own Arr_v, Arr_i, D_{LUT}, and $P_{c\text{-}Vdd}$ (see next). Notice cost and delay values of

mapping result to reduce potential node duplications as explained in Section 4.2. During the actual mapping, if some of the inputs of a cut are these tentative LUT roots, the cost of this cut will be recalculated and significantly reduced. The more tentative LUTs a cut's inputs contain, the larger the reduction of the cut's cost. This encourages LUT input sharing, i.e., a series of LUTs share the same input to reduce node duplications. After a cut is picked, its inputs are set as *actual* LUT roots for the later mapping process. Only nodes of actual LUT roots will be mapped iteratively. These actual LUT roots will join the tentative LUT roots for the later cut selection process, i.e., if some of the inputs of a cut are either tentative LUT roots or actual LUT roots, the cost of this cut will be recalculated. If a node v is on a critical path, only MC_v can be picked (see Section 4.1). If a node is on a non-critical path, the cut with smallest cost without timing violation is selected.

The mapping procedure is slightly more complicated than that for single Vdd because of the involvement of level converters. Suppose the relative delay numbers for V_H LUT, V_L LUT, and converter are 1, 1.4, and 0.3, respectively, Figure 7 illustrates a scenario. In (a), the right fanout of node R has two possible required times, depending on what kind of LUT node R will use. If R will use V_L, the dashed line is the critical path because there is a converter on the path, and 1.7 will be the correct required time for R ($1.7 = 3 - 1 - 0.3$). If R will use V_H, a required time of 2 will be propagated over from the right fanout, and the critical path will be on the left side (the required time for R will be 1.8). Consider another case shown in (b). Even when R uses V_L, the required time is 1.6 ($1.6 = 3 - 1.4$) and the critical path does not go through a converter. This shows that we need two special considerations to make the mapping procedure work correctly. First, we can use two types of required times for each node. One is for the case when R is using V_H, denoted as *req_time(R)*, and the other for V_L, denoted as *lvdd_req_time(R)*. Second, to accurately calculate *lvdd_req_time(R)*, we need to determine where the critical path is located. If the critical path goes through a converter, *lvdd_req_time(R)* deducts the converter delay from *req_time(R)*. Otherwise, it is equal to *req_time(R)*. Meanwhile, the *req_time* of fanins of R (x and y in the example) reflects the corresponding changes as well:

If R is using V_L:

$$req\_time(x \ or \ y) = lvdd\_req\_time(R) - D_{LUT\_VL};$$
$$= 1.7 - 1.4 = 0.3 \ \text{for case (a)}$$
$$= 1.6 - 1.4 = 0.2 \ \text{for case (b)}$$

If R is using V_H:

$$req\_time(x \ or \ y) = req\_time(R) - D_{LUT\_VH};$$

To map a node v, we go through the costs and delays of every cut rooted on v so that

$$P_{min} = \underset{\forall c \ on \ v}{\text{MIN}} \quad [\underset{\forall p \ on \ c}{\text{MIN}} \ P_{c\text{-}Vdd}] \qquad (11)$$

given the corresponding delay of P_{min} fulfills the following delay requirement:

Case 0 will join the delay and cost selection as well. It only provides V_H solutions.

benchmarks	Emap				SVmap			
	nodes	conn	est'ed power (w)	real power (w)	nodes	Conn	est'ed power (w)	real power (w)
alu4	1400	4556	0.1178	0.1336	1318	4432	0.1171	0.1340
apex2	1779	5641	0.1142	0.1658	1658	5514	0.1112	0.1518
apex4	1294	4136	0.0903	0.1068	1251	4179	0.0915	0.1118
bigkey	1818	6206	0.2220	0.1547	1709	6121	0.2043	0.1401
clma	7185	23191	0.7077	0.6662	7055	24025	0.7262	0.5920
des	1391	4717	0.2003	0.2253	1423	4753	0.1990	0.2351
diffeq	1070	3562	0.0636	0.0507	1077	3600	0.0658	0.0529
dsip	1374	5232	0.1644	0.1510	1372	5227	0.1643	0.1385
elliptic	2319	7685	0.1631	0.1715	2300	7773	0.1600	0.1683
ex1010	4405	14202	0.2635	0.3198	4286	14401	0.2599	0.3219
ex5p	1058	3357	0.1021	0.0867	1011	3563	0.1024	0.1030
frisc	2563	8429	0.1394	0.1863	2528	8600	0.1392	0.1854
misex3	1324	4137	0.1050	0.1186	1261	4083	0.1045	0.1245
pdc	4500	13997	0.2467	0.3956	4180	14103	0.2375	0.4040
s298	1738	6128	0.1142	0.0944	1651	5980	0.1099	0.0977
s38417	5624	17167	0.5605	0.4087	5211	17125	0.5430	0.3670
s38584	4944	15618	0.3909	0.3527	4671	15370	0.3843	0.3404
seq	1605	5043	0.1093	0.1505	1484	4961	0.1064	0.1467
spla	3727	12138	0.2337	0.2984	3584	12228	0.2289	0.3303
tseng	813	2534	0.0671	0.0378	803	2576	0.0666	0.0398
Average	2596.6	8383.8	0.2088	0.2137	2491.7	8430.7	0.2061	0.2093
Diff. %					**-4.0%**	**0.6%**	**-1.3%**	**-2.1%**

Table 4: Comparison details of SVmap and Emap

$$D_{Pmin} \leq req\_time(v) \qquad \text{if } V_{Pmin} \text{ is } V_H$$

$$D_{Pmin} \leq lvdd\_req\_time(v) \qquad \text{if } V_{Pmin} \text{ is } V_L$$

The cut with P_{min} is picked to implement the LUT on this node.[5] The LUT uses the voltage V_{Pmin}. The procedure continues until all the *PIs* are reached.

5. EXPERIMENTAL RESULTS
We will show the comparison results between the dual-Vdd mapping algorithm and the single-Vdd mapping algorithm to examine how technology mapping will affect FPGA power consumption with dual-Vdd considerations. We implement a single-Vdd mapper, *SVmap*. SVmap follows the delay and power propagation procedure as shown in Section 4.1, uses the cost function in Section 4.2, and relaxes non-critical paths to pick cuts with smaller cost. All the LUTs have the same delay under a 1.3v single Vdd. On the other hand, dual-Vdd settings use V_H as 1.3v and V_L as 0.8v, 0.9v or 1.0v. We call our dual-Vdd mapper *DVmap*.

To evaluate the effectiveness of our cost function and mapping procedure, we first compare *SVmap* with the latest published algorithm *Emap* [10], which is the state-of-the-art low-power single-Vdd mapper. Table 4 shows that *SVmap* offers some advantages over Emap in terms of area and power.[6] The power columns contain data for *estimated power* and *real power*. The estimated power column lists the power reported after mapping based on the power model presented in Section 3.2. The real

power column lists the power values obtained through our power estimator available in *fpgaEva_LP* [15], which reports power after placement and routing when actual routing capacitance and circuit delay values are available. We observe that our estimated power is very close to the real power. This gives us confidence that our power model is reasonably accurate. We also compare SVmap with another low-power FPGA mapper published in [8]. We use the 29 combinational benchmarks provided by the authors of [8]. SVmap shows 1.9% better area, 1.3% better connections, and 2.3% better power on average. The area gain of SVmap over [8] is smaller compared to the gain over Emap, because we run greedy pack on both SVmap and the mapper of [8] after mapping.

Table 5 lists all the power comparisons of the mapping results under different dual-Vdd combinations against our single-Vdd mapper. The combination of V_H as 1.3v and V_L as 0.8v offers the best power saving of an average of 11.6%. In the lower part of Figure 8, a bar chart of the power comparisons among these dual-Vdd combinations is shown.

The upper part of Figure 8 shows the ratio of number of V_L LUTs over total LUTs in our mapping results. For 1.3v-0.8v, the ratio is the smallest because the larger delay penalty of the 0.8v LUTs prevents more nodes on the non-critical paths from using V_L LUTs. On the other hand, the ratio for 1.3v-1.0v is the largest because of the small delay penalty of 1.0v LUTs. However each 1.0v LUT does not save as much power as a 0.8v LUT. This intuitively explains why 1.3v-0.8v gives the best results among the three. Table 6 shows the details for the case of the 1.3v-0.8v setting. We can observe that there are cases where the percentages of the V_L-LUT usages are very small. To better understand this scenario, we collect some details of *0-network* using SVmap. The 0-network consists of all the nodes that are on critical paths (slack 0) after mapping. We call these nodes critical LUTs. Table 7 shows the details. We observe that the larger percentage of critical LUTs over the total LUTs for a circuit, the smaller the number of

---

[5] All the cuts that do not fulfill the delay requirement are not considered here.

[6] We use the same benchmarks as Emap, which come with their own switching activities. Emap's switching activity calculation is based on the transition density model presented in [20].

	SVmap	DVmap		
benchmarks	v1.3	v1.3-v0.8	v1.3-v0.9	v1.3-v1.0
alu4	0.1171	0.1124	0.1111	0.1116
apex2	0.1112	0.1094	0.1101	0.1098
apex4	0.0915	0.0871	0.0864	0.0865
bigkey	0.2043	0.2050	0.2048	0.2048
clma	0.7262	0.6689	0.6727	0.6783
des	0.1990	0.1890	0.1828	0.1855
diffeq	0.0658	0.0501	0.0523	0.0547
dsip	0.1643	0.1649	0.1648	0.1647
elliptic	0.1600	0.1104	0.1170	0.1271
ex1010	0.2599	0.2574	0.2570	0.2570
ex5p	0.1024	0.1016	0.1019	0.1026
frisc	0.1392	0.1050	0.1086	0.1155
misex3	0.1045	0.0997	0.1009	0.0992
pdc	0.2375	0.2371	0.2364	0.2363
s298	0.1099	0.0935	0.0954	0.0959
s38417	0.5430	0.4539	0.4579	0.4600
s38584	0.3843	0.2306	0.2503	0.2756
seq	0.1064	0.1021	0.1027	0.0991
spla	0.2289	0.2174	0.2175	0.2157
tseng	0.0666	0.0471	0.0496	0.0530
Average	0.2061	0.1821	0.1840	0.1866
Diff %		**-11.6%**	**-10.7%**	**-9.4%**

Table 5: Dual-Vdd mapping results comparing with SVmap

Benchmarks	V_L LUTs	Total LUTs	Active Converters	V_L/Total Ratio
alu4	158	1319	134	12.0%
apex2	150	1644	148	9.1%
apex4	37	1192	36	3.1%
bigkey	4	1701	3	0.2%
clma	2108	7068	1271	29.8%
des	266	1304	100	20.4%
diffeq	770	1053	80	73.1%
dsip	1	1365	0	0.1%
elliptic	1816	2195	51	82.7%
ex1010	541	4178	539	12.9%
ex5p	88	1018	47	8.6%
frisc	2203	2438	102	90.4%
misex3	194	1235	178	15.7%
pdc	632	4163	408	15.2%
s298	958	1646	523	58.2%
s38417	2247	5106	274	44.0%
s38584	3881	4554	140	85.2%
seq	242	1482	181	16.3%
spla	588	3513	460	16.7%
tseng	654	803	28	81.4%
alu4	158	1319	134	12.0%

Table 6: Details of V_L LUT, active converter, and the V_L-LUT/Total-LUT ratio for 1.3v-0.8v Vdd setting after DVmap

V_L LUTs that can be accommodated for the circuit in Table 6. It is easy to see that the sum of percentages of V_L-LUT/Total-LUT and Critical-LUT/Total-LUT for each circuit will be ≤ 1.

6. CONCLUSION

We presented a cut enumeration algorithm targeting low-power technology mapping for FPGA architectures with dual supply voltages. We used a detailed delay and power model for LUTs of different voltages and level converters. The power model considered both dynamic power and static power of LUTs, converters, MUXes, and buffers. Detailed net wire capacitance was modeled as well. The algorithm built all the cases of LUT connections under dual-Vdd scenarios and generated one set of power and delay results for each case to enlarge the low-power solution search space. This is the first work of FPGA technology mapping targeting dual-Vdd architectures. Experimental results showed that we were able to save up to 11.6% of power consumption compared to the single-Vdd case. We also found that the 1.3v-0.8v dual-Vdd combination offered better power savings compared to the other two configurations.

7. ACKNOWLEDGMENTS

8. REFERENCES

[1] K. Usami and M. Horowitz, "Clustered Voltage Scaling for Low-Power Design," Intl. Sym. on Low Power Design, pp 3-8, April 1995.

[2] S. S. C. Yeh et al., "Gate Level Design Exploiting Dual Supply Voltages for Power-driven Applications," Proc. Design Automation Conference 1999, Jun. 1999.

[3] T. Mahnke, et al., "Efficiency of Dual Supply Voltage Logic Synthesis for Low Power in Consideration of Varying Delay Constraint Strictness," IEEE Intl. Conf. on Electronics, Circuits and Systems, Dubrovnik, Croatia, Sept. 2002.

[4] M. Hamada et al., "A Top-down Low Power Design Technique Using Clustered Voltage Scaling with Variable Supply-voltage Scheme," Proc. Custom Integrated Circuits Conference 1998, pp.495-498, May 1998.

[5] S. Raje and M. Sarrafzadeh, "Variable Voltage Scheduling," Intl. Sym. on Low Power Design, 1995.

[6] A.H. Farrahi and M. Sarrafzadeh, "FPGA Technology Mapping for Power Minimization," Proc. of Intl. Workshop in Field Programmable Logic and Applications, 1994.

[7] C.-Y. Tsui, M. Pedram, and A. M. Despain "Power Efficient Technology Decomposition and Mapping under an Extended Power Consumption Model," *IEEE TCAD*, pages 1110-1122, 1994.

[8] J. Anderson and F. N. Najm, "Power-Aware Technology Mapping for LUT-Based FPGAs," IEEE Intl. Conf. on Field-Programmable Technology, 2002.

[9] H. Li, W. Mak, and S. Katkoori, "Efficient LUT-Basd FPGA Technology Mapping for Power Minimization," ASPDAC 2003.

[10] J. Lamoureux and S.J.E. Wilton, "On the Interaction between Power-Aware CAD Algorithms for FPGAs," IEEE/ACM International Conference on Computer Aided Design, 2003.

[11] Xilinx, Virtex-II Pro Complete Data Sheet, Nov. 2003.

[12] Altera, Stratix Device Family Data Sheet, Nov. 2003.

[13] R. Puri et al., "Pushing ASIC Performance in a Power Envelope," Design Automation Conference, 2003.

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

[15] F. Li, D. Chen, L. He, and J. Cong, "Architecture Evaluation for Power-efficient FPGAs," ACM Intl. Sym. on FPGA, Feb. 2003.

[16] E. M. Sentovich et. al., "SIS: A System for Sequential Circuit Synthesis," Dept. of Elec. Engineering and Computer Science, UC Berkeley, CA 94720, 1992.

[17] G. Yeap, *Practical Low Power Digital VLSI Design*, Kluwer Academic Publishers, Boston, 1998.

[18] J. Cong and Y. Ding, "On Area/depth Trade-off in LUT-based FPGA Technology Mapping," DAC 1993.

[19] J. Cong, C. Wu and E. Ding, "Cut Ranking and Pruning: Enabling A General and Efficient FPGA Mapping Solution," Proc. ACM Intl. Symp. FPGA, February 1999.

[20] K.K.W. Poon, A. Yan, and S.J.E. Wilton, "A Flexible Power Model for FPGAs," 12th International Conference on Field-Programmable Logic and Applications, Sept. 2002.
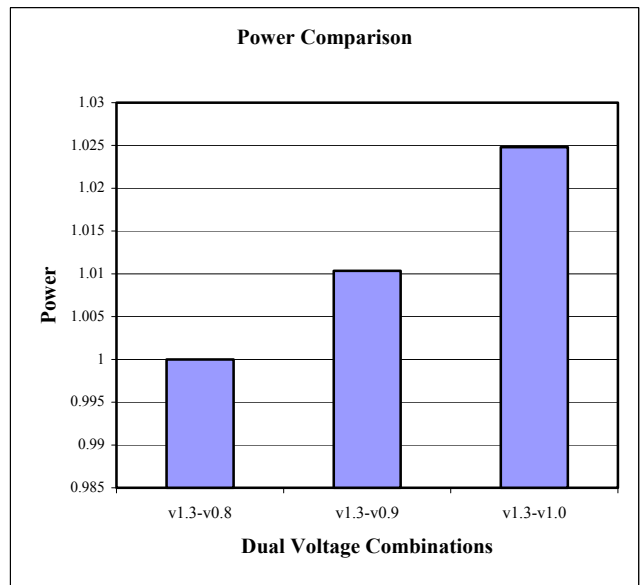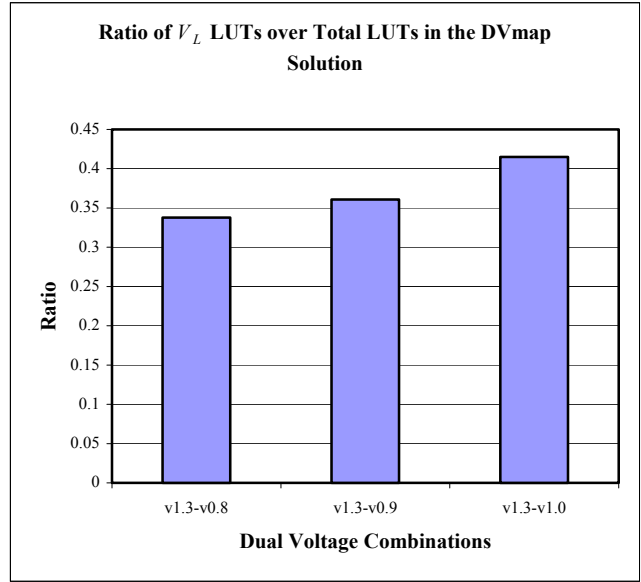
Figure 8: V_L/Total-LUT and power comparison for different dual-Vdd combinations

benchmarks	Total LUTs	Crit. LUTs	Crit./Total
alu4	1318	868	65.9%
apex2	1658	1098	66.2%
apex4	1251	1076	86.0%
bigkey	1709	1695	99.2%
clma	7055	3611	51.2%
des	1423	833	58.5%
diffeq	1077	143	13.3%
dsip	1372	1361	99.2%
elliptic	2300	210	9.1%
ex1010	4286	3068	71.6%
ex5p	1011	756	74.8%
frisc	2528	156	6.2%
misex3	1261	601	47.7%
pdc	4180	2211	52.9%
s298	1651	516	31.3%
s38417	5211	1750	33.6%
s38584	4671	258	5.5%
seq	1484	737	49.7%
spla	3584	2410	67.2%
tseng	803	84	10.5%

Table 7: Critical LUTs over total LUTs after SVmap