Technology Mapping and Clustering for FPGA Architectures with Dual Supply Voltages

Deming Chen, *Member, IEEE,* Jason Cong, *Fellow, IEEE,* Chen Dong, *Student Member, IEEE,* Lei He, *Senior Member, IEEE,* Fei Li, *Member, IEEE,* and Chi-Chen Peng

***Abstract*—This paper presents a technology mapping algorithm for field-programmable gate array 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. This paper also presents an enhanced clustering algorithm that considers dual supply voltages, honoring the dual-Vdd mapping solution. To carry out various comparisons, we first design a single-Vdd mapping algorithm, named *SVmap-2*, which achieves a 3.8% total power reduction (15.6% dynamic power reduction) over a previously published low-power mapping algorithm, *Emap* [11]. We then show that our dual-Vdd mapping algorithm, named *DVmap-2*, can further improve total power savings by 12.8% over *SVmap-2,* with a 52.7% dynamic power reduction. Compared to the early single-Vdd version *SVmap* [14], *DVmap-2* is 14.3% better for total power reduction. This is achieved through an ideal selection of the low-Vdd/high-Vdd ratio and the consideration of various voltage changing scenarios during the mapping process.**

***Index Terms*—Dual-supply voltages, field-programmable gate array (FPGA), power optimization, technology mapping.**

I. Introduction, Background, and Motivation

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 field-programmable gate array (FPGA) chips particularly, because FPGA chips are power inefficient compared to logically equivalent ASIC chips. The main reason is that FPGAs use a large number of transistors to provide field programmability. The large power consumption of FPGAs prevents FPGA designs from entering many low-power applications. The dynamic power of FPGAs is increasing significantly along the technology scaling. Therefore, reducing power consumption for FPGA chips is a critical task.

One of the popular design techniques for power reduction is to lower supply voltage, which results in a quadratic reduction of dynamic 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 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 works in [5] and [6] 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. These works motivated the multi-Vdd solution for low power and high performance, but none of them worked on FPGAs.

In this paper, we will study the power minimization problem at the logic synthesis level for FPGAs. Specifically, we will work on technology mapping and clustering for FPGA circuits using dual supply voltages. For *lookup table* (LUT)-based FPGAs, technology mapping converts a given Boolean circuit into a functionally equivalent network comprised only of LUTs. After technology mapping, clustering will group LUTs into logic clusters so that these clusters can be placed and routed on the FPGA chip.

The technology mapping for power minimization has been shown to be NP-complete to solve [7]. There are previous works on technology mapping for low-power FPGA designs, all assuming single Vdd [8]–[12]. Their basic approach was to hide the nodes of high-switching activity into LUTs so the overall dynamic power was reduced. Reference [29] presented a delay-optimal dual-Vdd clustering algorithm. However, due to aggressive duplication to guarantee delay optimality, the area overhead is large.

Manuscript received August 9, 2009; revised April 14, 2010 and May 28, 2010; accepted June 4, 2010. Date of current version October 20, 2010. This work was supported in part by NSF, under Grants CCR-0096383, CCR-0093273, CCR-0306682, and CCF 0746608, and by the Altera Corporation. This paper was recommended by Associate Editor J. Lach.

D. Chen, C. Dong, and C.-C. Peng are with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: dchen@illinois.edu; cdong3@illinois.edu; peng3@illinois.edu).

J. Cong is with the Department of Computer Science, University of California, Los Angeles, CA 90095 USA (e-mail: cong@cs.ucla.edu).

L. He is with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: lhe@ee.ucla.edu).

F. Li is with Actel Corporation, Mountain View, CA 94043 USA (e-mail: fei.li@actel.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2010.2061770

0278-0070/$26.00 © 2010 IEEE

In this paper, our main focus is to develop a low-power FPGA mapping algorithm, named *DVmap-2*, 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 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 also design a mapping algorithm with a single Vdd, named *SVmap-2*, which uses similar cost function as that in *DVmap-2* and relaxes the non-critical paths based on mapping cost to achieve better power results.

Specifically, *DVmap-2* and *SVmap-2* borrowed additional new techniques from [13] and applied them over the original algorithm *DVmap* and *SVmap* [14]. These techniques, namely, global duplication cost adjustment, input sharing, and slack distribution, have been helpful for area reduction during technology mapping in [13], and we use them here for the purpose of power reduction for both single-Vdd and dual-Vdd mapping. Then, we present an extended version of T-VPack [27] to support dual-Vdd clustering. The clustering algorithm takes the dual-Vdd mapping solution and produces clusters with different voltage assignments.

Compared to the original T-VPack, the area overhead of our dual-Vdd clustering is 1.2%. Dual-Vdd mapped and clustered solution is then passed to a power estimator *fpgaEva_LP2* [20], which supports dual-Vdd FPGA architectures. Experimental results show that *SVmap-2* is 3.8% better in terms of power reduction compared to a low-power mapping algorithm *Emap* [11]. We then show that *DVmap-2* can further improve *SVmap-2* by 12.8%. Compared to *SVmap* reported in [14], *DVmap-2* is 14.3% better. In addition, we show that *DVmap-2* can achieve a significant amount of dynamic power reduction over *SVmap-2*.

The rest of this paper is organized as follows. In Section II, we provide some basic definitions and formulate the dual-Vdd FPGA mapping and clustering problem. Section III introduces our FPGA architecture and power model. Section IV provides the detailed description of our algorithm. Section V presents experimental results, and Section VI concludes this paper.

II. 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 *(Y, Y')* of a cone C_v such that *Y'* is a cone of *v*. *v* is the *root* node of the cut. The node *cut-set* of the cut, denoted *V(Y, Y')*, consists of the inputs of cone *Y'*, or *input(Y')*. A cut is *K-feasible* if *Y'* is a *K*-feasible cone. In other words, the cardinality of the cut-set (or *cut size* of the cut) is = *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 $|input(v)|= K$ 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 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.

The dual-Vdd clustering problem is to take the solution of DVMF and cluster the LUTs into logic blocks where each logic block will be driven by a single voltage level while honoring the LUT's voltage assignment from the DVMF solution. Meanwhile, total amount of clusters and the critical path delay need to be reduced.

We assume that the input networks are all 2-bounded and *K* is 4 in this paper. 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 has been used among commercial FPGAs [15], [16] and is popular in academic studies. Our algorithm will work for any reasonable *K* values.

III. Architecture and Power Model

We will first introduce logic element and voltage level converter design to support dual Vdd in the FPGA. We then present our power model based on this architecture.

A. Logic Element and Level Converter

Fig. 1 shows the simplified model of the basic logic element (BLE) of a *K*-LUT-based FPGA. The output of the *K*-LUT can be either registered or unregistered. To guide the mapping process, we obtain the delay and power data of a 4-LUT for various supply voltages through SPICE simulation under $0.1\,\mu m$ technology. Table I 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

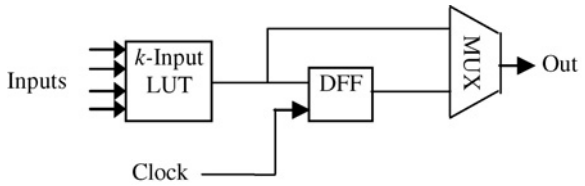


Fig. 1. Basic logic element.

TABLE I
DELAY AND POWER DATA FOR A 4-LUT

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

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. The power profile of LUT under different voltage levels has been pre-characterized and stored as a lookup table. These data will be used during the mapping process.

It has been shown that cluster-based logic blocks can improve the FPGA performance, area, and power [19], [20]. In this paper, a configurable logic block (CLB) can be driven by either a low supply voltage or a high supply voltage. Such Vdd configurability can be realized through a technique proposed in [17]. Fig. 2 shows the design. The basic idea is to insert two PMOS transistors between the high-Vdd (V_H) and low-Vdd (V_L) power rails and a CLB. The PMOS transistors are like sleep transistors, and the control bits C_1 and C_2 are used to control them so that an appropriate supply voltage can be chosen for the CLB. Fig. 3 shows the details of the CLB containing N BLEs. The inputs and outputs of a CLB can be programmed to go through level converters or bypass it. This gives us the capability to insert a level converter (LC) between a V_L CLB and a V_H CLB to handle the situation when a V_L CLB (*driver* CLB) is driving fanout CLBs (*end* CLB) of V_H voltage settings. A level converter is required when a V_L device output is to be connected to a V_H device input. Without such a converter, excessive leakage power would occur in the V_H device because the Vdd to Gnd path cannot be fully cut off due to the low input voltage.

With regard to voltage assignment for the routing tracks connecting the driver CLB and end CLBs, we follow a strategy proposed in [28]. The work [28] inserts level converters at CLB inputs and outputs and uses routing tree as the Vdd assignment unit. For a multi-fanout routing tree, only one voltage level will be assigned and this assignment is based on total power-sensitivity of all fanout pins. Power-sensitivity has been estimated based on switching activity and load capacitance. Routing tree with large capacitance and high switching activity will have large power-sensitivity, which will also have good potential of power saving if $V_L$is assigned. Routing tree based assignment can guarantee that there is no V_L interconnect switch drives V_H interconnect switch in the

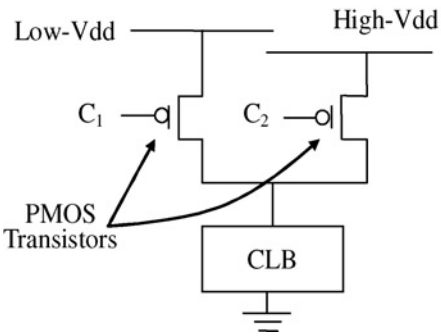


Fig. 2. CLB with Vdd configurability.

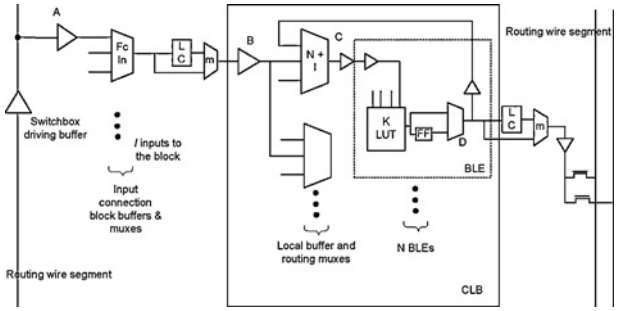


Fig. 3. CLB with inserted level converter.

routing so it does not need to insert level converters within the routing tracks as proposed in [21], which can incur large area overhead. When a V_L routing tree is driving V_H end CLBs (or when a V_L driver CLB is driving a V_H routing tree), the converters at the CLB inputs (outputs) will convert V_L to V_H. The routing switches are Vdd configurable through PMOS transistors. However, these PMOS transistors do not consume dynamic power after circuit is configured as they do not switch during normal operation. They do consume leakage power but high V_{th} (threshold voltage) and/or high T_{ox} (silicon dioxide thickness) can be applied to significantly limit leakage power without affecting circuit speed [28].

A MUX (m shown in Fig. 3) is inserted to bypass level converter when it is not needed. 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 0.014 ns, which is almost ignorable.

We use the level converter with single supply voltage as proposed in [18]. We show the transistor level schematic in Fig. 4. 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 (refer to [18] for details). Table II shows the detailed power 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.

B. 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

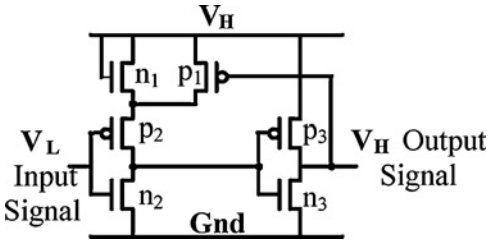


Fig. 4. Schematic of a level converter with single supply voltage.

TABLE II
DELAY AND POWER DATA FOR THE LEVEL CONVERTER

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

calculated as follows:

$$\begin{aligned} P_{cone} &= S_o \cdot P_{LUT_dynamic} + PLUT_static \\ &\quad + P_{inputs} + P_{net} \\ &= S_o \cdot (P_{LUT_dynamic} + P_{LUT_static}) \\ &\quad + (1 - S_o) P_{LUT_static} + P_{inputs} + P_{net} \\ &= S_o \cdot P_{LUT} + (1 - S_o) P_{LUT_static} + P_{inputs} + P_{net} \end{aligned} \quad (1)$$

where S_o is the switching activity of the cone output. $P_{LUT_dynamic}$ is the dynamic power of an LUT when the switching activity is 1. P_{LUT_static} is the static power of an LUT. P_{LUT} considers both dynamic and static power. P_{LUT} can be computed as $energy_per_switch * r$ (circuit frequency), where $energy_per_switch$ is reported in Table I. P_{inputs} is the power consumed on the cut inputs, which is defined as follows:

$$Pinputs = 0.5r \cdot Vdd^2 \sum_{i}^{k} Si \cdot Cin \quad (2)$$

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

$$P_{net} = 0.5r \cdot Vdd^2 \cdot C_{net} \cdot S_o + P_{buf_static} \quad (3)$$

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 different gate by gate. To obtain reasonable wire and buffer capacitance in the net before placement and routing, we profile a series of benchmarks using *VPR* [19] as the placement and routing tool.

Fig. 5 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. Fig. 6 shows the average wire length across 20 benchmarks for each fanout number when

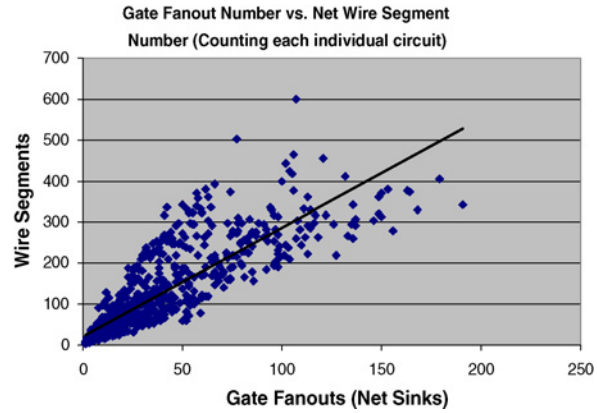


Fig. 5. Gate fanout number vs. wire length driven by the gate.

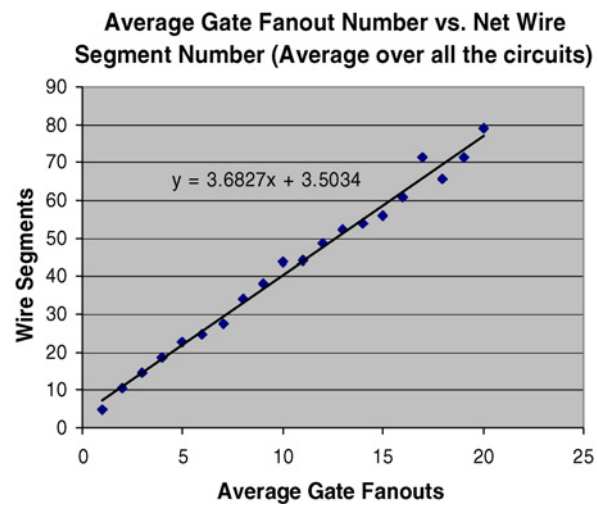


Fig. 6. Average gate fanout number vs. wire length for. smaller gate fanout numbers.

the fanout number is ≤ 20. The correlation can be considered as linear in Fig. 6. Since most of the gates have relatively small fanout numbers, we will use the plotted trend line in Fig. 6 to estimate the net capacitance on the gate output.

Both S_i and S_o are calculated up front before the mapping starts. We use the switching activity calculator available in SIS [22], which builds binary decision diagram (BDD) 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 node v is then calculated by a formula as $2 \cdot Pv \cdot (1 - Pv)$ [23], where Pv is the probability of node v being 1. Switching activities are inputs to our algorithm. Therefore, any switching activity estimation method can be used as long as it is sufficiently accurate.

IV. ALGORITHM DESCRIPTION

A. Overview

The overall CAD flow of this paper is shown in Fig. 7. The first stage is dual-Vdd technology mapping and can be divided into two major sub-steps: cut enumeration and cut selection. In general, technology mapping can be carried

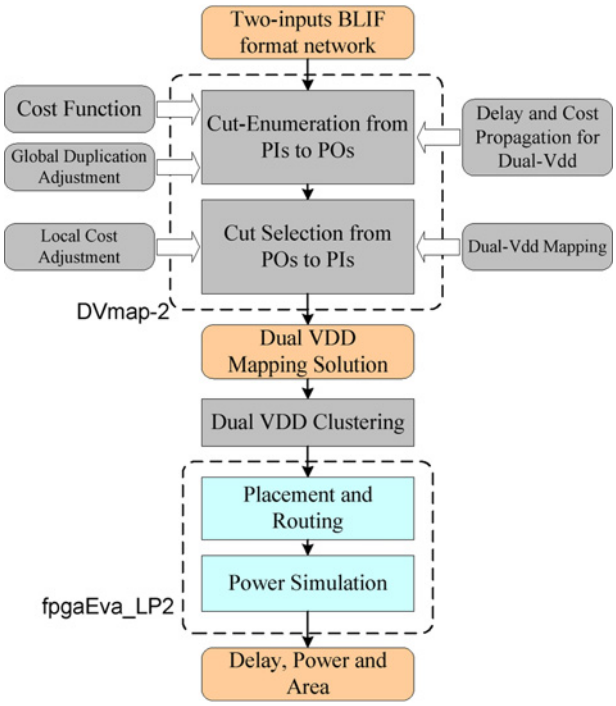


Fig. 7. CAD flow.

out through pattern matching and covering, where pattern matching identifies the logic cones rooted on a node that match the library cells, and covering will pick the actual library cells to cover the entire netlist. In FPGA, cut-enumeration is an effective method to find out all the possible ways of the K-feasible cones rooted on a node that match the library cell—the K-LUT. Both [24] and [25] used this method for mapping to minimize area. The works in [9], [11], and [12] present low-power mappers based on this technique as well. After cut enumeration and arrival time and cost propagation, a cut selection procedure is carried out to cover the entire netlist. This procedure is guided by the required time, which is the optimal mapping depth of the network determined during the cut enumeration process.

The second stage is the dual-Vdd clustering step, which has been adopted from T-VPack [27]. Dual-Vdd clustering takes the LUTs from the dual-Vdd mapping solution and clusters them into dual-Vdd CLBs for placement and routing. In the last stage of the flow, we use an FPGA evaluation framework *fpgaEva_LP2* [20] to perform placement and routing on the clustered netlists. Power simulator *Psim* in *fpgaEva_LP2* is used to measure power consumption.

Next, in Section IV-B, we introduce cut enumeration and the propagation of arrival time and power cost. Section IV-C introduces the cost computation for a cut itself, U_C. In Section IV-D, we enhance the power propagation estimation with a global cost adjustment technique for better duplication control. Section IV-E then extends the delay and cost computation into the dual-Vdd domain. Sections IV-F and IV-G present the cut selection procedure. Specifically, in Section IV-F, cut selection with local cost adjustment is presented so a better selection solution can be generated. Section IV-G then discusses the cut selection procedure with dual Vdd. The top portion of Fig. 7 illustrates the relationships among these tasks. Overall, Sections IV-B–IV-E talk about the matching part, and Sections IV-F and IV-G talk about the covering part of the mapping algorithm. We then introduce the dual-Vdd clustering algorithm in Section IV-H.

B. Cut Enumeration

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 $V(Y_v, Y_v')$. 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 [25]:

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

where $f(K, v)$ represents all the K-feasible cuts rooted at node v, operator + is *Boolean OR*, and $\otimes^K$ is *Boolean AND*. After (4) is expanded into a sum-of-product format, each resulting p-term is a possible cut. We will filter 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

$$Arr_v = \underset{\forall Con\ v}{\text{MIN}} \quad [\underset{i \in input(C)}{\text{MAX}} (Arr_i) + 1] \tag{5}$$

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. Notice that there can be more than one MC_v for node v, and all the MC_v's form a set X_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:

$$P_C = \sum_{i=input(c)} [P_i / f_i] + U_C \tag{6}$$

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. 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. This is similar to the idea present in [25] where area instead of power was estimated. However, since we do not know whether there will

be duplications for node i at this point, this model is still a heuristic.

To guarantee optimal mapping depth, we need to propagate the estimated power together with the propagation process of the minimum arrival time. Thus, P_i of (6) in our case will be the best propagated power in the fanin cone F_i. Then, after we have calculated the power for each cut rooted on node v, the best propagated power in the fanin cone F_v is as follows:

$$Pv = \underset{C \in Xv}{\text{MIN}} \ (P_C). \tag{7}$$

P_v represents the best achievable mapping power up to node v under the constraint that it also generates the optimal mapping delay up to the point of v (all the cuts considered belong to set X_v). This P_v is used as the P_i in (6) if v is an input of next cut. Through (6) and (7), the powers of the cuts and nodes are iteratively calculated until the enumeration process reaches all the POs. Later on, during the cut selection stage when we know that v is not on a critical path, a cut $C \notin X_v$ may be picked as long as it will not violate the timing constraint and will produce a better power in the same time (a relaxed arrival time provides more suitable cuts and a better opportunity for better power mapping). Thus, P_v represents a power estimation guarded by a timing constraint that usually is overstressed unless v is on a critical path. Nonetheless, we have to stick to P_v during cut enumeration because we do not know whether v is a critical node or not until the entire cut-enumeration process is completed. This motivates a better cut selection algorithm for better power saving.

After cut enumeration, we obtain the optimal mapping depth of the network. This mapping depth will be set as the required time for the circuit. The critical path(s) is the path that leads to such a mapping depth, and the nodes on non-critical paths will have the luxury of selecting different cuts that offer smaller costs (including those offered through a low Vdd) with a relaxed delay value as long as the required time of the circuit is maintained. More details are introduced in sections F and G.

C. Calculation of Cut Cost

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. 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. We will use an example to explain these factors.

1) *Node Coverage:* In Fig. 8, cuts 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.

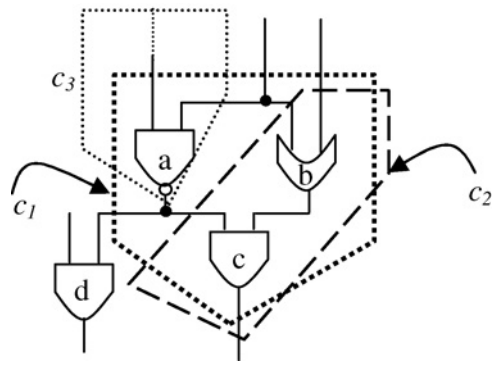


Fig. 8. Illustration of various cuts.

2) *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 [20], 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.

3) *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 is, the larger the chance that the cut will be picked during the cut selection stage. This naturally selects cuts that hide highly switching nodes into 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.

4) *Output Fanout Number:* Another factor we consider is the fanout number of the root node of the target cut. This is trying to control node duplication because duplication usually hurts power minimization [9]. In Fig. 8, C_1 contains the node a, which has a fanout that goes outside of C_1. This indicates that if $C_1$is picked, node a has to be duplicated in order to drive node d. On the contrary, if cut C_3 is picked, it can drive both node d and c directly so there is no need to duplicate node a. The larger the fanout number, the better for picking this cut, because it potentially saves more duplications. More papers about duplication modeling will be discussed in Section IV-C from a different angle.

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

$$U_C = \frac{CS_C(1 + \alpha \sum S_i)}{(1 + \alpha \cdot COV_C + \beta \cdot FT_C)}. \tag{8}$$

CS_C is the cut size of the cut C. S_i is the switching activity on input i of the cut. COV_C is the total number of nodes covered by the cut, and FT_C is the fanout number of the root node. α and β are constants.

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 III-B. It is intuitive that the quality of the mapping result

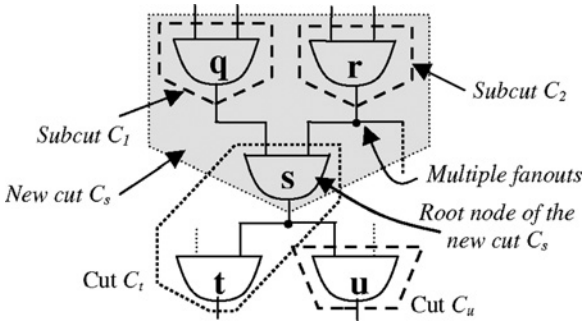


Fig. 9. Power cost adjustment.

is directly related to the accuracy of power estimation. In the next several sections, we will present some additional techniques to improve the accuracy of the power model. These techniques are categorized as global cost adjustment and local cost adjustment.

D. Global Duplication Cost Adjustment

The power model shown in (6) has the advantage of considering the branching effect on the fanout. However, it is not perfect. Using the example shown in Fig. 9, if cut C_s is used to implement an LUT in the final mapping and there is no duplication involved, the power rooted on node s, P_s, should be equally shared by fanouts t and u. Otherwise, P_s will be falsely double-counted when it is propagated to both t and u later. However, this estimation has its downside. Suppose the final mapping result uses C_t and C_u (Fig. 9), then the estimation is no longer accurate because C_u treats node s as not duplicated[1] but s is actually duplicated in C_t. Thus, the power model can under-estimate the actual mapping power. This issue can be dealt with during cut enumeration. We will adjust the estimated power according to the potential node duplication scenarios. These can be captured by checking the subcuts that are with or without multiple fanouts. In Fig. 9, when subcuts C_1 and C_2 are combined to form C_s, we observe that there will not be any node duplications for node q because it is a single-fanout node. However, there will be a duplicated node for node r since it has another fanout (dashed line) that goes out of cut C_s. We change (6) to the following:

$$P_C = \sum_{i=input(C)}[P_i/f_i] + U_C + P_{I_1} + P_{I_2} \tag{9}$$

where I_1 and I_2 are the two fanin nodes of the root node v, and cut C is formed by the two subcuts C_{I_1} and C_{I_2} rooted on I_1 and I_2 respectively (e.g., C_s is formed by C_1 and C_2 in the example). P_{I_1} and P_{I_2} are the duplication costs of the subcuts, which are defined respectively as follows:

$$P_I = \begin{cases} N_{CI}/CSc & \text{if } f_I > 1 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where C_I is the subcut on either I_1 or I_2, and f_I is the fanout number of corresponding I_1 or I_2. N_{CI} is the number of nodes C_I contains, and CS_C is the cutsize of C. The intuition is that the larger N_{CI} is, the larger the possibility that more nodes in C_I will be duplicated, thus the larger duplication cost it produces. CS_C is treated as a normalizing factor because the larger CS_C is, the more likely it is that C will contain more nodes. This, in a way, alleviates the duplication cost experienced in the local area of the cut, i.e., in C_I. Notice, once P_C is adjusted it stays that way, and the related node cost will also consequently be adjusted. Meanwhile, the new cost will propagate to the cuts and nodes on the fanouts. In Fig. 9, once the costs of C_s and P_s are adjusted, they will start to influence the costs of C_u and P_u, etc. Thus, this cost adjustment has a global impact for the power propagation process and makes the power estimation more closely related to the actual mapping implementation in a global point of view.

[1]Ps is divided by 2 and propagated to Cu.

TABLE III
DUAL-VDD SCENARIOS

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

E. Delay and Cost Propagation for Dual Vdd

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

During cut enumeration, beside the delay and cost value calculated for the single-Vdd situation, each cut will have additional power and delay values corresponding to the four cases listed in Table III. 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

$$Arr_v = \underset{\forall C \text{ on } v}{\text{MIN}} \; [\underset{i \in input(C)}{\text{MAX}} (Arr_i) + D_{LUT} + \{D_{conv}\}] \tag{11}$$

where $[\text{MAX}\ (Arr_i) + D_{LUT} + \{D_{conv}\}]$ is the arrival time for a cut C rooted on v. Here, 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. Arr_i is influenced by LUT_1's voltage setting, in this case, V_L. D_{LUT} is 1 in this case because LUT_2 is using V_H. [2] 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 due to the *case*1 and *case*3 scenario in the previous delay propagation (when v was the root node of LUT_1). These two cases provided Arr_i values with the V_L setting in the previous propagation. We will pick the case that gives smaller MAX (Arr_i) value for the current calculation, and its cost is used for

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

cost propagation. If these two cases provide the same delay, the case with smaller cost will be picked for cost propagation.[3] At the end, each node would contain four additional solution points corresponding to cases 1 to 4. All of these solution points will be propagated to the next level as what *case* 0 does. That way, the benefit of low Vdd on power can be incorporated and evaluated. We introduce how the cost computation for the cut next.

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

$$P_{C-Vdd} = \sum_{i=input(C)}[P_i/f_i] + U_{C-Vdd} + \{U_{conv}\} + P_{I1} + P_{I2}. \quad (12)$$

P_i is the propagated cost on input i with LUT_1's voltage setting. P_{I1} and P_{I2} are the duplication costs of the subcuts (Section IV-C). U_{C-Vdd} is the cost of C (LUT_2) itself. It can be either U_{C-VL} or U_{C-VH}, depending on LUT_2's voltage setting. The value of U_{C-VL} is computed in a similar way as the one defined for single Vdd, U_C. U_{C-VH} is proportionally larger than U_{C-VL} as follows:

$$U_{C-VH} = (Power_{C-VH}/Power_{C-VL}) \bullet U_{C-VL} \quad (13)$$

where $Power_{C-VH}$ and $Power_{C-VL}$ are estimated 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 III-B. This gives an accurate proportional increase of U_{C-VH} over U_{C-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. At the end, each cut would contain four additional solution points corresponding to cases 1 to 4.

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-Vdd} will become Arr_i and P_i for next iteration during the calculation.

F. Cut Selection with Local Cost Adjustment

The difficulty of mapping lies in the method of selecting cuts to cover the whole circuit to minimize the total power. We cannot greedily pick the cuts with the smallest costs calculated through the cut-enumeration process, because that will forfeit some important optimization factors in terms of reducing node duplications locally. We introduce the details next.

To map a critical node v, only the cut that provides P_v (7) is picked to implement the LUT to guarantee the optimal mapping depth. How to select the cuts for the non-critical nodes thus becomes the key to reduce power. Local cost adjustment can increase the chance of duplication reduction during final mapping. We will not simply pick the cut with the best cost calculated by the cut-enumeration process with global duplication cost adjustment. Instead, the cost of the examined cut will be adjusted depending on the characteristics of the cut itself (thus the term *local*). After the cost adjustment, it is possible that another cut with an originally unfavorable cost becomes the most favorable to map the current node. We will introduce two techniques below.

1) *Input Sharing:* During the cut selection procedure, we try to pick a cut for a node. We will check to see if some of the cut-set nodes (input nodes) of a cut C are already LUT roots. If this is the case, C will not generate as many new LUT roots as other cuts do when those other cuts do not have this feature.[4] In other words, cut C takes advantage of existing resources and does not require new resources. This reduces the chances that a newly picked cut will cut into the interior territories of existing LUTs. As a result, the input nodes are shared among several mapped LUTs, and node duplications are reduced. The cost of cut C is recalculated and significantly reduced according to how many inputs it shares from existing LUT roots. Thus, the cut selection is largely influenced by the local settings around the target node and its cuts.

2) *Slack Distribution:* We define the slack on a node v as follows:

$$Slack_v = Req_v - Arr_v \quad (14)$$

where the required time for v, Req_v, is defined as follows:

$$Req_v = \underset{v \in input(LUT_I)}{MIN}(Req_i - 1) \quad (15)$$

where Req_i is the required time of LUT i that has v as an input.

When $Slack_v$ is greater than 0, it means v is not on the critical path so that there is more flexibility to choose a cut C that is not in X_v, as long as the required time propagated back to the input nodes of this cut is still larger or equal to the minimum arrival times on those nodes. It is easy to see that if C uses up all the slacks available on v, there will exist at least one path in F_v that will no longer have the flexibility to pick cuts outside of X_n, where n is on that path and $n \neq v$. So, we want to distribute the slacks along the edges of the entire paths to encourage more nodes on the paths to have the flexibility to search their solution space. We design a simple slack distribution technique, which is applied in terms of the adjusted cost. We define the slack of a cut C rooted on v as follows:

$$Slack_C = Req_v - 1 - \underset{i \in input(C)}{\text{MAX}}(Arr_i). \quad (16)$$

If $Slack_C < 0$, C is not a *timing_feasible* cut. Choosing it will violate the optimal mapping depth constraint, so such a cut will be discarded. The larger the $Slack_C$, the better for C in terms of slack distribution effects. We then adjust the cost of C accordingly.

All of the local adjustment techniques are implemented in the subroutine *pick_cut*. Its high-level description is shown in Fig. 10. *share_no* is the number of shared inputs of a cut C

[3]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-Vdd}. Notice cost and delay values of Case 0 will join the delay and cost selection as well. It only provides V_H solutions.

[4]Suppose a node w is a cut-set node for both cut C_a rooted on node a and cut C_b rooted on b. C_a is picked to implement LUT_a, and w becomes a future LUT root. Later on, when we map b, C_b has an advantage because it shares input w with LUT_a.

```
Algorithm pick_cut
input: target node v, output: LUT_v.
if v is critical, return the cut that provides P_v;
best_cost = ∞;
for each timing_feasible cut C on v do
    cost_C := P_C;
    share_no := number of shared input nodes of C;
    if share_no = 0 then
        share_no := 1;
    else-if share_no = 1 then
        share_no := 1.15;
    end-if
    cost_C := cost_C / share_no;
    cost_C := cost_C − ε · Slack_C;
    if (best_cost > cost_C) then
        best_cost := cost_C;
        LUT_v := C;
    end-if
end-for
output LUT_v;
```

Fig. 10. Algorithm of *pick_cut*.

with existing LUTs. The new cost is the old cost divided by *share_no*. When *share_no* is 1, we set it as 1.15 (obtained by empirical study) to count in the sharing effect. The direct effect of sharing an input with an existing LUT is that the portion of area from the input can be ignored for the target cut. Division achieves this purpose by weighing each shared input with the same amount of cost savings. It is a local operation concentrating on edge reduction. We use the term $\varepsilon \cdot Slack_C$ to count the slack distribution effect. Symbol ε is 0.3 in our case.

Based on these techniques, we developed a single-Vdd mapping algorithm, named *SVmap-2*, which demonstrates better results than a previous low-power mapping algorithm as shown in the experimental result section. Next, we will extend *SVmap-2* to *DVmap-2* to consider dual supply voltages.

G. Dual-Vdd Mapping Generation

To support dual-Vdd, 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. Local cost adjustment described in Section IV-F is applied during the mapping process.

The mapping procedure is 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, Fig. 11 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

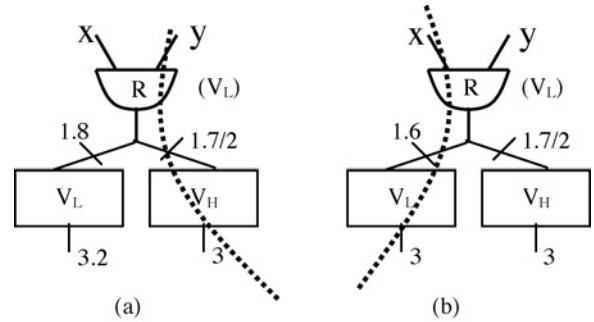


Fig. 11. Critical path and level converter delay. Numbers are required times. (a) Converter on critical path. (b) Converter not on critical path.

in (b). Here, 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

$$\begin{aligned} req_time(x \text{ or } y) &= lvdd_req_time(R) - D_{LUT_VL}; \\ &= 1.7 - 1.4 = 0.3 \quad \text{for case (a)} \\ &= 1.6 - 1.4 = 0.2 \quad \text{for case (b).} \end{aligned} \tag{17}$$

If R is using V_H

$$req_time(x \text{ or } y) = req_time(R) - D_{LUT_VH}. \tag{18}$$

To map a node v, we go through the delay of every solution point of every cut rooted on v so that the delay of the solution point fulfills the following delay requirement:

$$D_{sol} \leq req_time(v) \quad \text{if } V_{sol} \text{ is } V_H \tag{19}$$

$$D_{sol} \leq lvdd_req_time(v) \quad \text{if } V_{sol} \text{ is } V_L. \tag{20}$$

We then go through a cut selection procedure similar to Fig. 9. The cut selected would use the corresponding voltage V_{sol}. The procedure continues until all the PIs are reached.

The worst case complexity of our *DVmap-2* is $O(n^K)$ where K is the LUT input size and n is the total number of nodes in the network. However, similar to DAOmap in [13], when LUT input K is small, the number of cuts generated for each node v is a small constant because all the cuts are usually generated within a small cone. In average the complexity of the cut-enumeration algorithm is practically linear to n for small K ($K < 7$) [13]. Table VI lists the run time for both *SVmap-2* and *DVmap-2*.

TABLE IV
COMPARISON DETAILS OF SVMAP-2 AND EMAP

	Emap			SVmap-2		
Benchmarks	LUT No.	Connect No.	Power (W)	LUT No.	Connect No.	Power (W)
alu4	1400	4556	0.2076	1295	4559	0.2061
apex2	1779	5641	0.2253	1627	5631	0.2410
apex4	1294	4136	0.1612	1167	3977	0.1504
bigkey	1818	6206	0.6960	1707	6117	0.6614
clma	7185	23 191	0.6884	6858	23 827	0.6652
des	1391	4717	0.8438	1286	4703	0.8027
diffeq	1070	3562	0.1053	1082	3650	0.0999
dsip	1374	5232	0.6807	1370	5449	0.6405
elliptic	2319	7685	0.2463	2216	7697	0.2216
ex1010	4405	14 202	0.4211	4137	14 104	0.4148
ex5p	1058	3357	0.1248	969	3318	0.1268
frisc	2563	8429	0.2261	2363	8450	0.2074
misex3	1324	4137	0.1930	1202	4137	0.1898
pdc	4500	13 997	0.3805	4099	14 054	0.3858
s298	1738	6128	0.1650	1702	6176	0.1534
s38417	5624	17 167	0.5555	4931	16 702	0.5464
s38584	4944	15 618	0.5342	4539	15 528	0.5068
seq	1605	5043	0.2246	1449	5014	0.2246
spla	3727	12 138	0.3399	3453	12 230	0.3095
tseng	813	2534	0.1031	811	2727	0.1007
Average	2596.6	8383.8	0.3561	2413.2	8402.5	0.3427
Diff. %				**−7.1%**	**0.2%**	**−3.8%**

H. Dual-Vdd Clustering

T-VPack [27] is an efficient tool to pack LUTs and registers into the logic clusters, but it only supports single Vdd. The original T-VPack minimizes the cluster number by packing as many BLEs as possible into a CLB. In the meantime, it also minimizes the number of external connections (connections between clusters) on the critical path and then reduces the critical path delay. However, to generate the dual-Vdd clustering solution, we need to add a new constraint. We first review the key ideas of T-VPack. Then we introduce our new changes.

1) *Overview of T-VPack:* T-VPack has three optimization goals. The first is to pack each logic block to its capacity in order to minimize the number of blocks needed (each block is a CLB). The second goal is to minimize the number of inputs to each block in order to reduce the number of connections required between blocks. The last is to minimize the number of external connections (connections between blocks) on the critical path. The first two goals can be achieved through the following attraction function:

$$Attraction(B) = Nets(B) \cap Nets(C) \tag{21}$$

where BLE B is packed into block C based on the *Attraction* value, which is determined by the number of inputs and outputs that B and C have in common.

To achieve the last goal, T-VPack extends the above attraction function to include timing information. The first BLE that is placed into a cluster (i.e., block) is the unclustered BLE that is on the critical path. Then, T-VPack adds the most attractive BLEs to the cluster based on the new attraction function to be presented below. It repeats these processes until either no more BLEs will fit into the cluster, or all of the cluster inputs are used. Once a cluster is full, it starts a new cluster with a new seed, and repeats the process until there are no unclustered BLEs left in the circuit. The extended attraction function considers *Base_BLE_Criticality* and *Total_Paths_Affected*. *Base_BLE_Criticality(B)* is defined to be the maximum *Connection_Criticality* value of all connections joining B (the unclustered BLE) to BLEs within the cluster currently being packed. *Total_Paths_Affected* includes *Input_Paths_Affected* and *Output_Paths_Affected*. And the new attraction function is defined as [27]

$$\begin{aligned} Attraction(B) = \alpha \cdot (&Base_BLE_Criticality(B) \\ &+\varepsilon \cdot total_\ paths_affected(B)) \\ &+(1-\alpha) \cdot \frac{Nets(B) \cap Nets(C)}{G} \end{aligned} \tag{22}$$

where ε is a very small value that ensures that the *Total_Paths_Affected* value acts only as a tie-breaking mechanism. G is a normalization factor which is set to the maximum number of nets to which any BLE can connect. α is a tradeoff variable determining how the attraction to be affected by criticality vs. input pin sharing.

2) *New Constraint for Dual Vdd:* The goal of dual-Vdd clustering is to pack the BLEs with the same voltage level into the same cluster, while at the same time still keeping all optimization goals of T-VPack. This can be done by adding a new constraint into the packing procedure. When we try to pack the most attractive BLE into a cluster, it checks whether the voltage level of this BLE is the same as the voltage level of the cluster. The voltage level of the cluster is determined by the voltage level of the seed or the first BLE in this cluster. If the seed is a latch or flip-flop, we can check the voltage level of the second or other nodes until we run into a BLE. Then, the voltage of the cluster is determined by the voltage of this BLE. This also means that if the node which T-VPack

TABLE V
POWER COMPARISONS OF *DVmap*-2 AGAINST *SVmap* AND *SVmap*-2

Benchmark	SVmap	SVmap-2	DVmap-2		DVmap-2		DVmap-2	
	v1.3	v1.3	v1.3-v0.8		v1.3-v0.9		v1.3-v1.0	
	Power (W)	Power (W)	Power (W)	Vs. Svmap2	Power (W)	Vs. Svmap2	Power (W)	Vs. Svmap2
alu4	0.2021	0.2061	0.1669	−19.03%	0.1685	−18.22%	0.1736	−15.75%
apex2	0.2202	0.2410	0.1958	−18.76%	0.1984	−17.67%	0.2036	−15.52%
apex4	0.1620	0.1504	0.1293	−14.04%	0.1299	−13.68%	0.1330	−11.57%
bigkey	0.6917	0.6614	0.5194	−21.47%	0.5288	−20.05%	0.5451	−17.59%
clma	0.6739	0.6652	0.5928	−10.89%	0.5919	−11.02%	0.6032	−9.31%
des	0.8377	0.8027	0.6699	−16.55%	0.6774	−15.61%	0.6892	−14.15%
diffeq	0.1019	0.0999	0.0845	−15.46%	0.0845	−15.37%	0.0858	−14.12%
dsip	0.6677	0.6405	0.5005	−21.87%	0.5119	−20.08%	0.5266	−17.79%
elliptic	0.2382	0.2216	0.2030	−8.37%	0.2038	−7.99%	0.2085	−5.90%
ex1010	0.4204	0.4148	0.3816	−7.98%	0.3817	−7.98%	0.3884	−6.34%
ex5p	0.1196	0.1268	0.1155	−8.94%	0.1152	−9.19%	0.1167	−7.96%
frisc	0.2403	0.2074	0.2014	−2.90%	0.2005	−3.33%	0.2039	−1.69%
misex3	0.1857	0.1898	0.1550	−18.29%	0.1567	−17.44%	0.1603	−15.53%
pdc	0.3735	0.3858	0.3601	−6.66%	0.3589	−6.97%	0.3654	−5.28%
s298	0.1572	0.1534	0.1478	−3.64%	0.1451	−5.40%	0.1486	−3.10%
s38417	0.5198	0.5464	0.4825	−11.69%	0.4855	−11.14%	0.4906	−10.22%
s38584	0.5413	0.5068	0.4197	−17.18%	0.4273	−15.69%	0.4433	−12.53%
seq	0.2167	0.2246	0.1877	−16.42%	0.1889	−15.91%	0.1913	−14.85%
spla	0.3484	0.3095	0.2905	−6.14%	0.2899	−6.32%	0.2955	−4.53%
tseng	0.1013	0.1007	0.0917	−8.89%	0.0928	−7.86%	0.0942	−6.38%
Ave.	0.3510	0.3427	0.2948	**−12.8%**	0.2969	**−12.4%**	0.3033	**−10.5%**
DVmap-2* vs. *SVmap			**−14.3%**		**−13.9%**		**−12.1%**	

tries to pack is a latch or flip-flop, we would directly pack it into the cluster. Therefore, latches or flip-flops will be set to either high or low Vdd determined by the voltage level of the cluster itself.

Finally, the modified T-VPack outputs the voltage level of each cluster and the clustered netlist for *fpgaEva_LP2* to carry out placement and routing and power evaluation. The runtime and complexity of dual-Vdd packing is similar to those of the original T-VPack.

V. EXPERIMENTAL RESULTS

As mentioned before, we name our single-Vdd mapping algorithm *SVmap-2* and dual-Vdd mapping algorithm *DVmap-2*. *SVmap-2* follows the delay and power propagation procedure as shown in Section IV-B, uses the cost function in Sections IV-C–IV-D, and relaxes non-critical paths to pick cuts with smaller cost augmented with the local cost adjustment. All the LUTs have the same delay under a 1.3v single Vdd. On the other hand, dual-Vdd settings for *DVmap-2* use V_H as 1.3v and V_L as 0.8v, 0.9v or 1.0v. We carry out various types of comparison studies. The first is to evaluate the quality of *SVmap-2* compared to a previously published single-Vdd mapping algorithm, *Emap* [11]. The second is between *DVmap-2* and *SVmap*/*SVmap-2* to show how dual Vdd would help to reduce power consumption. Next, we evaluate our dual-Vdd clustering algorithm. Then, we carry out dynamic power comparisons. Finally, to evaluate how much improvement this paper has obtained over the conference versions [14], we compare *DVmap-2* to *DVmap*.

To obtain post placement and routing power consumption values for both the single-Vdd and the dual-Vdd cases, an FPGA evaluation platform *fpgaEva_LP2* [20] has been used in this paper to implement placement, routing, capacitance/delay extraction and power estimation. 2000 random input vectors have been generated in *fpgaEva_LP2* for individual benchmark to carry out gate-level power simulation. A 5% delay overhead of CLB due to Vdd programmability (PMOS tran-

TABLE VI
LUT PERCENTAGE IN *DVmap*-2 VS. THE CRITICAL LUT PERCENTAGE IN *SVmap*-2 AND THE RUN TIME (IN SECOND) OF *DVmap*-2 AND *SVmap*-2

Benchmarks	V_L/Total (DVmap-2)	Crit./Total (SVmap-2)	Run time (DVmap-2)	Run time (SVmap-2)
alu4	15.3%	65.8%	0.52	0.17
apex2	9.1%	79.7%	0.42	0.17
apex4	2.9%	91.9%	0.36	0.15
bigkey	0.2%	99.2%	0.5	0.24
clma	33.9%	48.9%	4.81	1.62
des	30.3%	58.9%	0.44	0.21
diffeq	61.9%	19.8%	0.39	0.19
dsip	0.2%	99.1%	0.51	0.24
elliptic	77.3%	9.3%	1.76	0.62
ex1010	12.3%	70.8%	1.66	0.74
ex5p	12.4%	73.6%	0.31	0.14
frisc	58.9%	6.9%	1.28	0.56
misex3	18.5%	54.9%	0.37	0.16
pdc	10.9%	48.1%	2.2	0.9
s298	58.8%	40.4%	1.8	0.58
s38417	46.5%	29.1%	3.56	1.5
s38584	85.5%	6.2%	13.79	4.02
seq	19.1%	48.5%	0.45	0.2
spla	12.1%	62.8%	1.67	0.71
tseng	79.4%	9.4%	0.36	0.14

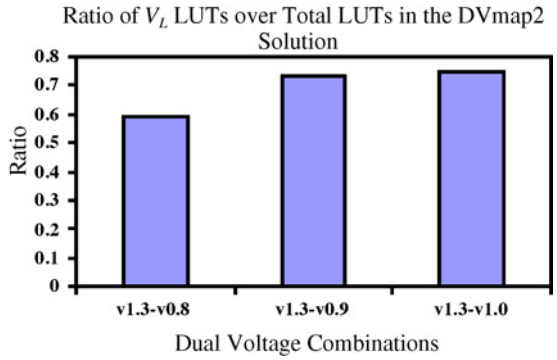


Fig. 12. V_L/Total-LUT for different dual-Vdd combinations.

sistors in Fig. 2) has been set in the dual-Vdd mode. Due to the area increment, a 6% interconnect delay overhead has been considered as well. Both numbers have been suggested by [20], [28] for 0.1um technology using CLB-based Vdd assignment. An FPGA architecture with a channel width of 100 has been selected.

A. Comparison Between SVmap-2 and Emap

Emap [11] is a low-power single-Vdd mapper that uses cut enumeration-based technique as well. Its mapping procedure is similar to what is presented in Section IV-B although it uses a different cost function. The cost function tries to reduce node duplication, switching activity and LUT connections. It reported 7.6% better power than an area optimization-oriented mapping algorithm using 4-LUT ($K = 4$).

Table IV shows that *SVmap-2* offers advantages over *Emap* in terms of area and total power. Its area is 7.1% smaller on average compared to *Emap*. The power values are obtained through the power estimator *fpgaEva_LP2* for the single-Vdd case. *SVmap-2* on average consumes 3.8% less total power compared to *Emap* after placement and routing. In terms of dynamic power, *SVmap-2* achieves a 15.6% dynamic power reduction (Table VIII).

B. Comparison between DVmap-2 and Single-Vdd Solutions

Table V lists all the post placement and routing power comparisons of the mapping results under dual-Vdd against *SVmap* (the mapper in the conference version [14]) and *SVmap-2*. The combination of V_H as 1.3v and V_L as 0.8v offers the best power savings of an average of 14.3% and 12.8% over *SVmap* and *SVmap-2*, respectively.

Fig. 12 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. 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 does. This intuitively explains why 1.3v–0.8v gives the best results among the three in Table V. In Table VI, column 2 shows the percentage of the V_L LUTs out of the total LUTs for each benchmark for the 1.3v–0.8v setting. We can observe that for some cases the percentages of the V_L-LUT usage are very small, and low Vdd does not provide much advantage for low power in these cases. To better understand this scenario, we collect some details of the 0-network using *SVmap-2*. The 0-network consists of all the nodes that are on critical paths (slack 0) after mapping. We call these nodes critical LUTs. Column 3 of Table VI shows the percentage of the critical LUTs out of the total LUTs for each benchmark reported from *SVmap-2*. We observe that the larger the percentage of critical LUTs for a circuit, the smaller the percentage of V_L LUTs that can be accommodated for the circuit. This is easy to understand because our dual-Vdd mapping still guarantees the minimum mapping delay, and only non-critical nodes can use V_L cells.

TABLE VII
COMPARISON DETAILS OF ORIGINAL AND MODIFIED T-VPACK

Benchmarks	Orig. T-VPack	Dual-Vdd T-VPack			V_L/Total (DVmap-2)	Area Area Overhead
	Total Cluster	Total Cluster	V_L Cluster	V_L/Total		
alu4	133	135	17	12.6%	15.3%	1.5%
apex2	167	170	16	9.4%	9.1%	1.8%
apex4	124	126	5	4.0%	2.9%	1.6%
bigkey	171	172	1	0.6%	0.2%	0.6%
clma	708	716	211	29.5%	33.9%	1.1%
des	132	132	27	20.5%	30.3%	0.0%
diffeq	107	107	78	72.9%	61.9%	0.0%
dsip	131	139	1	0.7%	0.2%	6.1%
elliptic	225	227	187	82.4%	77.3%	0.9%
ex1010	442	445	57	12.8%	12.3%	0.7%
ex5p	108	110	11	10.0%	12.4%	1.9%
frisc	249	249	224	90.0%	58.9%	0.0%
misex3	126	128	20	15.6%	18.5%	1.6%
pdc	422	428	67	15.7%	10.9%	1.4%
s298	166	167	96	57.5%	58.8%	0.6%
s38417	551	553	248	44.8%	46.5%	0.4%
s38584	480	484	400	82.6%	85.5%	0.8%
seq	152	154	26	16.9%	19.1%	1.3%
spla	355	360	61	16.9%	12.1%	1.4%
tseng	81	82	66	80.5%	79.4%	1.2%
Average	**251.5**	**254.2**	**91**	**33.8%**	**32.3%**	**1.2%**

C. Comparison of original and modified T-VPack

Table VII shows that we have a similar ratio of low Vdd clusters over total clusters (Column 5) to the ratio of low-Vdd LUTs to total LUTs (Column 6) in *DVmap-2*. Also, we only have a 1.2% area overhead compared to the original T-VPack in terms of total number of CLBs (Column 7). The area overhead is reasonable since we added one additional constraint for dual-Vdd packing, which reduces the flexibility of packing for CLB number reduction.

D. Dynamic power comparisons

Table VIII lists the dynamic power reduction of *DVmap-2* under different voltage settings over *Emap* and *SVmap-2*. We can observe that dual Vdd is an effective technique for dynamic power reduction. *DVmap-2* offers up to 52.7% dynamic power reduction over *SVmap-2* on average. The savings are much more significant compared to the total power reduction because leakage power is dominating in the architecture model of *fpgaEva_LP2*. Modern commercial FPGAs adopted advanced process technologies (e.g., triple-oxide, dual V_{th} transistors, etc.) to reduce leakage power [30].

TABLE VIII
COMPARISON OF DYNAMIC POWER CONSUMPTION ACROSS DIFFERENT MAPPING ALGORITHM

	Emap	SVmap2	DVmap2 v1.3-0.8	DVmap2 v1.3-0.9	DVmap2 V1.3-1.0
alu4	0.105	0.101	0.060	0.063	0.067
apex2	0.103	0.101	0.054	0.058	0.061
apex4	0.060	0.049	0.027	0.028	0.030
bigkey	0.259	0.225	0.104	0.105	0.120
clma	0.164	0.126	0.042	0.046	0.072
des	0.252	0.211	0.098	0.106	0.117
diffeq	0.019	0.011	0.006	0.007	0.008
dsip	0.250	0.208	0.080	0.093	0.106
elliptic	0.061	0.035	0.014	0.016	0.019
ex1010	0.079	0.070	0.030	0.034	0.037
ex5p	0.040	0.041	0.029	0.031	0.034
frisc	0.039	0.018	0.009	0.010	0.011
misex3	0.092	0.087	0.051	0.053	0.056
pdc	0.065	0.064	0.032	0.034	0.037
s298	0.040	0.028	0.020	0.022	0.023
s38417	0.145	0.130	0.059	0.066	0.067
s38584.1	0.182	0.151	0.056	0.068	0.081
seq	0.105	0.103	0.065	0.067	0.068
spla	0.077	0.043	0.017	0.020	0.022
tseng	0.022	0.018	0.009	0.011	0.012
Avg	0.108	0.091	0.043	0.047	0.052
vs. Emap	**0.0%**	**−15.6%**	**−60.1%**	**−56.6%**	**−51.5%**
vs. SVmap2		**0.0%**	**−52.7%**	**−48.4%**	**−42.9%**

Thus, dynamic power is still the dominating portion in the total power. For such devices, dual-Vdd mapping and clustering will produce more significant total power reduction.

E. Improvement of DVmap-2/SVmap-2 over [14]

We also performed a comparison between *DVmap-2* and the original *DVmap* [14] for the v1.3-v0.8 case. Details are omitted due to the page limit. *DVmap-2* is 0.6% better on LUT number. However, *DVmap-2* increases the number of low-Vdd LUTs by 4.3% (level converters increase accordingly). Overall, *DVmap-2* is, on average, 2.2% better than *DVmap* in terms of power consumption. This shows that the new techniques added (Sections IV.D and IV.F) help to improve the mapping results in general. *SVmap-2* on average is 2.1% better than *SVmap* in terms of power consumption (Table V).

VI. CONCLUSION

We presented a cut enumeration-based algorithm targeting low-power technology mapping for FPGA architectures with dual supply voltages. We first designed an effective cost function for single-Vdd mapping, and then we extended it to consider dual-Vdd cases. We used a detailed delay and power model for LUTs and level converters of different voltages. 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. We also presented a dual-Vdd clustering algorithm adopted from the algorithm T-VPack. Experimental results showed that we were able to save up to 12.8% of total power compared to the new single-Vdd algorithm *SVmap-2* and 14.3% of total power compared to the original algorithm *SVmap*. In terms of dynamic power, we were able to achieve a 52.7% reduction over *SVmap-2*. We also found that the 1.3v–0.8v dual-Vdd combination offered better power savings compared to the other voltage configurations.

REFERENCES

[1] K. Usami and M. Horowitz, "Clustered voltage scaling for low-power design," in *Proc. Int. Symp. Low Power Design*, Apr. 1995, pp. 3–8.

[2] S. S. C. Yeh, M.-C. Chang, S.-C. Chang, and W.-B. Jone, "Gate level design exploiting dual supply voltages for power-driven applications," in *Proc. Design Automat. Conf.*, 1999, pp. 68–71.

[3] T. Mahnke, S. Panenka, M. Embacher, W. Stechele, and W. Hoeld, "Efficiency of dual supply voltage logic synthesis for low power in consideration of varying delay constraint strictness," in *Proc. Int. Conf. Electron., Circuits Syst.*, Sep. 2002, pp. 701–704.

[4] M. Hamada, M. Takahashi, H. Arakida, A. Chiba, T. Terazawa, T. Ishikawa, M. Kanazawa, M. Igarashi, K. Usami, and T. Kuroda, "A top-down low power design technique using clustered voltage scaling with variable supply-voltage scheme," in *Proc. Custom Integr. Circuits Conf.*, May 1998, pp. 495–498.

[5] S. Raje and M. Sarrafzadeh, "Variable voltage scheduling," in *Proc. Int. Symp. Low Power Design*, 1995, pp. 9–14.

[6] D. Chen, J. Cong, and J. Xu, "Optimal simultaneous module and multi-voltage assignment for low-power," *ACM Trans. Design Automat. Electron. Syst.*, vol. 11, no. 2, pp. 362–386, Apr. 2006.

[7] A. H. Farrahi and M. Sarrafzadeh, "FPGA technology mapping for power minimization," in *Proc. Int. Workshop Field Program. Logic Applicat.*, 1994, pp. 66–77.

[8] C.-Y. Tsui, M. Pedram, and A. M. Despain, "Power efficient technology decomposition and mapping under an extended power consumption model," *IEEE Trans. CAD*, vol. 13, no. 9, pp. 1110–1122, Sep. 1994.

[9] J. Anderson and F. N. Najm, "Power-aware technology mapping for LUT-based FPGAs," in *Proc. Int. Conf. Field-Program. Technol.*, 2002, pp. 211–218.

[10] H. Li, W. Mak, and S. Katkoori, "Efficient LUT-basd FPGA technology mapping for power minimization," in *Proc. Asia South Pacific Design Automat. Conf.*, 2003, pp. 353–358.

[11] J. Lamoureux and S. J. E. Wilton, "On the interaction between power-aware CAD algorithms for FPGAs," in *Proc. Int. Conf. Comput.-Aided Design*, 2003, pp. 701–708.

[12] L. Cheng, D. Chen, and D. F. Wong, "GlitchMap: An FPGA technology mapper for low power considering glitches," in *Proc. Design Automat. Conf.*, Jun. 2007, pp. 318–323.

[13] D. Chen and J. Cong, "DAOmap: A depth-optimal area optimization mapping algorithm for FPGA designs," in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 2004, pp. 752–759.

[14] D. Chen, J. Cong, F. Li, and L. He, "Low-power technology mapping for FPGA architectures with dual supply voltages," in *Proc. Int. Symp. FPGA*, Feb. 2004, pp. 109–117.

[15] Altera [Online]. Available: http://www.altera.com/products/prd-index.html

[16] Xilinx [Online]. Available: http://www.xilinx.com/products/index.htm

[17] F. Li, Y. Lin, and L. He, "FPGA power reduction using configurable dual-Vdd," in *Proc. Design Automat. Conf.*, Jun. 2004, pp. 735–740.

[18] R. Puri, L. Stok, J. Cohn, D. Kung, D. Pan, D. Sylvester, A. Srivastava, and S. Kulkarni, "Pushing ASIC performance in a power envelope," in *Proc. Design Automat. Conf.*, 2003, pp. 788–793.

[19] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Norwell, MA: Kluwer, Feb. 1999.

[20] F. Li, Y. Lin, L. He, D. Chen, and J. Cong, "Power modeling and characteristics of field programmable gate arrays," *IEEE Trans. CAD*, vol. 24, no. 11, pp. 1712–1724, Nov. 2005.

[21] F. Li, Y. Lin, and L. He, "Vdd programmability to reduce FPGA interconnect power," in *Proc. Int. Conf. Comput.-Aided Design*, Nov. 2004, pp. 760–765.

[22] E. M. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and

A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis," Dept. Elec. Eng. Comput. Sci., Univ. California, Berkeley, Mem. UCB/ERL M92/41 94720, 1992.

[23] G. Yeap, *Practical Low Power Digital VLSI Design*. Boston, MA: Kluwer, 1998.

[24] J. Cong and Y. Ding, "On area/depth trade-off in LUT-based FPGA technology mapping," in *Proc. Design Automat. Conf.*, 1993, pp. 213–218.

[25] J. Cong, C. Wu, and E. Ding, "Cut ranking and pruning: Enabling a general and efficient FPGA mapping solution," in *Proc. Int. Symp. FPGA*, Feb. 1999, pp. 29–35.

[26] K. K. W. Poon, A. Yan, and S. J. E. Wilton, "A flexible power model for FPGAs," in *Proc. Int. Conf. Field-Programm. Logic Applicat.*, Sep. 2002, pp. 312–321.

[27] A. Marquardt, V. Betz, and J. Rose, "Using cluster-based logic blocks and timing-driven packing to improve FPGA speed and density," in *Proc. Int. Symp. FPGAs*, 1999, pp. 37–46.

[28] Y. Lin, F. Li, and L. He, "Power modeling and architecture evaluation for FPGA with novel circuits for Vdd programmability," in *Proc. Int. Symp. FPGA*, Feb. 2005, pp. 199–207.

[29] D. Chen and J. Cong, "Delay optimal low-power circuit clustering for FPGAs with dual supply voltages," in *Proc. Int. Symp. Low Power Electron. Design*, 2004, pp. 70–73.

[30] M. Klein. (2009, Apr. 13). "Power consumption at 40 and 45 nm." *White Paper* [Online]. Available: http://www.xilinx.com/support/documentation/white_papers/wp298.pdf

Deming Chen (S'01–M'05) received the B.S. degree in computer science from the University of Pittsburgh, Pittsburgh, PA, in 1995, and the M.S. and Ph.D. degrees in computer science from the University of California, Los Angeles, in 2001 and 2005, respectively.

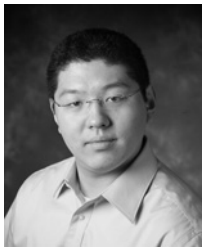
He was a Software Engineer from 1995 to 1999 and from 2001 to 2002. He was an Assistant Professor with the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign (UIUC), Urbana, since 2005. His current research interests include nano-systems design and nano-centric CAD techniques, field-programmable gate array synthesis and physical design, high-level synthesis, microarchitecture and system-on-a-chip design under parameter variation, and reconfigurable computing.

Dr. Chen is a technical committee member for a series of conferences and symposia. He is an Associated Editor for TVLSI, TCAS-I, JCSC, and JOLPE. He received the Achievement Award for Excellent Teamwork from Aplus Design Technologies in 2001, the Arnold O. Beckman Research Award from UIUC in 2007, the NSF CAREER Award in 2008, the ASPDAC'09 Best Paper Award, the SASP'09 Best Paper Award, and the ACM SIGDA Outstanding New Faculty Award in 2010. He is included in the List of Teachers Ranked as Excellent in 2008.

Jason Cong (S'88–M'90–SM'96–F'00) received the B.S. degree in computer science from Peking University, Beijing, China, in 1985, and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign, Urbana, in 1987 and 1990, respectively.

Currently, he is a Chancellor's Professor with the Department of Computer Science, University of California, Los Angeles (UCLA), the Director of Center for Domain-Specific Computing (funded by the NSF Expeditions in Computing Award), a Co-Director of UCLA/Peking University Joint Research Institute in Science and Engineering, and a Co-Director of the VLSI CAD Laboratory, UCLA. He also served as the Department Chair of UCLA Computer Science Department from 2005 to 2008. His current research interests include computer-aided design of VLSI circuits and systems, design and synthesis of system-on-a-chip, programmable systems, novel computer architectures, nano-systems, and highly scalable algorithms. He has published over 300 research papers and led over 30 research projects in these areas.

Dr. Cong has received a number of awards and recognitions, including the Ross J. Martin Award for Excellence in Research from the University of Illinois at Urbana-Champaign in 1989, the NSF Young Investigator Award in 1993, the Northrop Outstanding Junior Faculty Research Award from UCLA in 1993, the ACM/SIGDA Meritorious Service Award in 1998, and the SRC Technical Excellence Award in 2000. He has also received five Best Paper Awards. He was elected as an ACM Fellow in 2008.



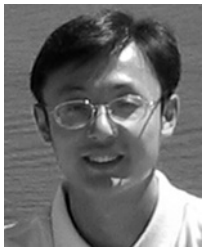
Chen Dong (S'09) received the B.S. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 2004, and the M.S. degree in electrical and computer engineering from Indiana University-Purdue University Indianapolis, Indianapolis, in 2006. Since 2006, he has been pursuing the graduate degree in electrical and computer engineering from the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana.

His current research interests include non-traditional/non-silicon devices, CMOS/nano hybrid reconfigurable high performance and low power computing, process variation aware 3-D physical design flow of field-programmable gate arrays, and device/circuit/architecture exploration and optimization for nanoscale very large scale integration.

Lei He (M'99–SM'08) received the Ph.D. degree in computer science from the University of California, Los Angeles, in 1999.

He is currently a Professor with the Department of Electrical Engineering, University of California. He was a Faculty Member with the University of Wisconsin-Madison, Madison, from 1999 to 2002. He held visiting or consulting positions with Cadence, Santa Clara, CA, Empyrean Soft, Beijing, China, Hewlett-Package, Palo Alto, CA, Intel, Santa Clara, CA, and Synopsys, Mountain View, CA, and was a technical advisory board member for Apache Design Solutions, Mountain View, CA, and Rio Design Automation, Santa Clara, CA. His current research interests include modeling and simulation, very large scale integration circuits and systems, and cyber physical systems.

Dr. He has published one book and over 200 technical papers with 12 Best Paper nominations mainly from the Design Automation Conference and International Conference on Computer-Aided Design and five Best Paper or Best Contribution Awards, including the 2010 Best Paper Award from ACM Transactions on Design Automation of Electronic Systems.



Fei Li (M'02) received the B.S. degree in electrical engineering from Fudan University, Shanghai, China, in 1997, the M.S. degree in computer engineering from the University of Wisconsin-Madison, Madison, in 2002, and the Ph.D. degree in electrical engineering from the University of California, Los Angeles, in 2005.

He is currently with Actel Corporation, Mountain View, CA. His current research interests include programmable device architecture, low-power design, and computer-aided design of very large scale integration circuits and systems.

Chi-Chen Peng received the B.A.Sc. degree in electrical engineering from National Taiwan Ocean University, Keelung, Taiwan, in 1997, and the M.S. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign, Urbana, in 2010.

In 1999, he joined Winbond, Inc., Hsinchu, Taiwan, as a SRAM Testing Engineer, where he has been involved in SRAM fault detection and yield improvement. In 2002, his responsibility was expanded to Flash and became a SRAM/Flash Testing Engineer.