Dual-Vdd Interconnect With Chip-Level Time Slack Allocation for FPGA Power Reduction

Yan Lin, *Student Member, IEEE*, and Lei He, *Member, IEEE*

***Abstract*—To reduce field-programmable gate array power, Vdd programmability has been recently proposed to select the Vdd level for interconnects and power-gate unused interconnects. However, Vdd-level converters used in the existing Vdd-programmable method consume a large amount of leakage. This paper proposes two ways to avoid using level converters in interconnects, namely; 1) tree-based level converter insertion (TLC) and 2) dual-Vdd tree-based level converter insertion (dTLC). TLC enforces that there is only one Vdd level within each routing tree, while dTLC can have different Vdd levels within a routing tree, but no VddL switch drives VddH switches. Dual-Vdd assignment algorithms were developed considering chip-level time slack allocation for maximum power reduction. The algorithms include TLC-S and dTLC-S, two power sensitivity-based algorithms with implicit time slack allocation, and dTLC-LP, a linear programming (LP)-based algorithm with explicit time slack allocation. All allocate time slack first to interconnects with higher power sensitivity and assign low Vdd to them for more power reduction. Experiments show that dTLC-LP obtains the lowest power consumption. Compared to dTLC-LP, dTLC-S obtains a slightly higher power consumption but runs three times faster. Compared to the existing segment-based level converter insertion for dual Vdd, dTLC-LP reduces interconnect power by 52.90% without performance loss for Microelectronics Center of North Carolina benchmark circuits.**

***Index Terms*—Interconnect, low-power design, optimization, power minimization.**

I. Introduction

FIELD-PROGRAMMABLE gate array (FPGA) power modeling and reduction has become an active research area recently. Poon *et al.* [1] and Li *et al.* [2] present power evaluation frameworks for generic parameterized FPGA architectures and show that both interconnect and leakage powers are significant for nanometer FPGAs. Mukherjee and Memik [3] present a power-driven partition algorithm for mapping applications to FPGA with different Vdd levels. Lamoureux and Wilton [4] study the interaction of a suite of power-aware FPGA CAD algorithms without changing the existing FPGAs. Anderson *et al.* [5] propose a configuration inversion method to reduce the leakage power of multiplexers without any additional hardware. In addition, low-power FPGA circuits and architectures have been proposed. Srinivasan *et al.* [6] design a new type of routing multiplexer and propose an input control method to reduce unused routing multiplexer leakage. Gayasen *et al.* [7] develop region-based power gating for unused FPGA logic blocks to reduce leakage power. Lodi *et al.* [8] study low leakage interconnect circuitry, which combines gate biasing, body biasing, and multithreshold techniques to reduce interconnect leakage. Li *et al.* [9], [10] are the first to introduce the dual Vdd and field programmability of Vdd to FPGA. Lin *et al.* [11] apply fine-grained power gating to FPGA interconnects and present a routing algorithm for predetermined dual-Vdd interconnects. Vdd programmability has been applied to both FPGA logic blocks [9], [10] and interconnects [12]–[14].

In this paper, we aim to improve the Vdd-programmable interconnects proposed in [13], where a Vdd-level converter is inserted in front of each interconnect switch to avoid excessive leakage when a low-Vdd (VddL) interconnect switch drives high-Vdd (VddH) interconnect switches. A level converter is also inserted at each logic block (CLB) input and output. This can be referred as segment-based level converter insertion (SLC). It provides fine-grained Vdd programmability for interconnects and may help to maximize dynamic power reduction. However, SLC also introduces large leakage and area overhead. As presented in Table I, with power gating of unused interconnect switches [13], [15], the average leakage and area due to level converters in routing channels contribute 38.73% of the total power and 33.96% of the total area, respectively, for the 20 largest Microelectronics Center of North Carolina (MCNC) benchmark circuits [16] at the ITRS 100-nm technology node [17]. Power gating unused level converters may reduce leakage but introduce an extra area overhead.[1] Therefore, it is less attractive compared to the methods to be presented in this paper to minimize the number of level converters.

Two existing approaches avoid directly using level converters in Vdd-programmable interconnects. Anderson and Najm [14] use level-restore buffers, where an NMOS power transistor is used to generate the VddL level by leveraging the threshold voltage drop of the NMOS transistor when passing "1." However, the range of VddL is limited by the threshold voltage of the NMOS power transistor. The positive-feedback PMOS transistor in the level-restore buffer is in fact an alternative level converter, which has less leakage overhead but may cause larger delay and therefore larger short circuit power compared to the level converter used in [13]. Gayasen *et al.* [12] only insert a

Manuscript received December 24, 2004; revised June 1, 2005 and September 8, 2005. This work was supported in part by the National Science Foundation under CAREER Award CCR-0093273 and Grant CCR-0306682. This paper was recommended by Associate Editor E. Macii.

The authors are with the Department of Electrical Engineering, University of California, Los Angeles, CA 90034 USA (e-mail: lhe@ee.ucla.edu).

Digital Object Identifier 10.1109/TCAD.2006.870858

[1]Our experiments show that by using a 1X PMOS transistor as a power transistor power gating can reduce the leakage power of an unused level converter by 1000× with 8% level converter delay increase and 3% chip-level area overhead at the ITRS 100-nm technology node.

TABLE I
LEAKAGE POWER AND AREA OF Vdd-LEVEL CONVERTERS (LCS) IN VDD-PROGRAMMABLE INTERCONNECTS WITH SLC USING 100-nm TECHNOLOGY. AREA IS IN NUMBER OF MINIMUM WIDTH TRANSISTORS

circuit	total power (watt)	total area	# of LCs	leakage of LCs		area of LCs	
				power (watt)	% of total power	area	% of total area
alu4	0.09563	7136109	122980	0.01771	18.52%	2341017	32.81%
apex2	0.11226	11078412	201536	0.02902	25.85%	3841352	34.67%
apex4	0.06435	7534174	137020	0.01973	30.66%	2616818	34.73%
bigkey	0.19473	19628149	285740	0.04115	21.13%	5366894	27.34%
clma	0.50864	100634278	2050655	0.29529	58.06%	39102935	38.86%
des	0.20639	28245400	415715	0.05986	29.00%	7810165	27.65%
diffeq	0.03603	7667824	128921	0.01856	51.5 2%	2449969	31.95%
dsip	0.20503	22792498	357175	0.05143	25.09%	6737844	29.56%
elliptic	0.11073	23394478	438283	0.06311	57.00%	8348076	35.68%
ex1010	0.16727	30506887	584995	0.08424	50.36%	11148195	36.54%
ex5p	0.06304	7808090	143065	0.02060	32.6 8%	2733309	35.01%
frisc	0.19169	46672107	944892	0.13606	70.98%	18038234	38.65%
misex3	0.08475	7548212	132440	0.01907	22.5 0%	2523180	33.43%
pdc	0.24585	43487207	878867	0.12656	51.48%	16781015	38.59%
s298	0.05599	9485182	157652	0.02270	40.5 5%	2991013	31.53%
s38417	0.26492	40827249	743341	0.10704	40.41%	14117128	34.58%
s38584	0.23410	33144130	602095	0.08670	37.04%	11438246	34.51%
seq	0.11620	11078412	201536	0.02902	24.98%	3841352	34.67%
spla	0.15882	25585907	498311	0.07176	45.18%	9509592	37.17%
tseng	0.03072	5397201	88660	0.01277	41.56%	1685087	31.22%
avg.					38.73%	-	33.96%

level converter at each CLB input or output. All the routing trees driven by (driving) a CLB have the same Vdd level as the source (sink) CLB when level converters are inserted at CLB inputs (outputs). A path-based assignment is performed to assign a Vdd level for CLBs and interconnects. However, this path-based assignment lacks flexibility and cannot effectively leverage the surplus timing slack.

If CAD algorithms can guarantee that no VddL interconnect switch drives VddH switches, no level converter is needed. The primary contribution of this paper is to remove level converters in routing channels by developing novel Vdd-level assignment algorithms to guarantee that no VddL switch drives VddH switches. We propose tree-based level converter insertion (TLC), where there is only one Vdd level within each routing tree [see Fig. 1(a)]. We also propose dual-Vdd tree-based level converter insertion (dTLC), where different Vdd levels can exist within a routing tree, but no VddL switch drives VddH switches [see Fig. 1(b)]. In both cases, configurable level converters are inserted at inputs and outputs of logic blocks (CLBs) to allow the mix of Vdd levels for CLBs and interconnects.

Our second contribution is to propose a few Vdd-level assignment algorithms considering time slack allocation to maximize power reduction. For the specified clock cycle time $T_{\rm spec}$, path timing constraints can be translated into delay upper bounds for nets using delay budgeting. A greedy algorithm [18] and a convex programming technique [19] have been developed for delay budgeting in placement. Yeh and Marek-Sadowska [20], [21] apply delay budgeting to minimize flip-flop number and improve performance in sequential applications of FPGA. In this paper, we represent the net delay upper bound by time slack, i.e., the amount of delay that could be added to routing trees without increasing $T_{\rm spec}$. We then allocate time slack to each routing tree. The allocated slack allows VddL switches and reduces interconnect power.

Our Vdd-level assignment algorithms include two power sensitivity-based algorithms, TLC-S and dTLC-S for TLC and dTLC problems, respectively, and a linear programming (LP)-based algorithm, dTLC-LP for the dTLC problem. TLC-S and dTLC-S implicitly allocate time slack first to interconnects with larger power sensitivity and then assign VddL to them for more power reduction. dTLC-LP first explicitly allocates time slack to each routing tree by formulating the problem as an LP problem to maximize a lower bound of power reduction, and then the Vdd-level assignment is solved optimally within each routing tree given the allocated time slack. Experiments show that dTLC-LP obtains the lowest power consumption among all the formulations and algorithms. Compared to dTLC-LP, dTLC-S obtains a slightly higher power consumption but runs three times faster. The best algorithm, dTLC-LP, reduces the total interconnect power by 52.90% compared to the SLC [13] and by 18.52% compared to the best approach proposed in [12].

The rest this paper is organized as follows. Section II introduces modeling and problem formulation. Section III describes power sensitivity-based algorithms and the LP-based algorithm. Section IV presents the experimental results, and Section V concludes this paper. The preliminary results of TLC were first presented in [15].[2]

II. MODELING AND PROBLEM FORMULATION

A. Preliminaries

Interconnects consume most of the area and power of FPGAs. We assume the traditional island-style routing architecture [22] as shown in Fig. 1 in our paper. The logic blocks (CLBs) are surrounded by routing channels consisting of wire segments. We use CLBs with cluster size $N = 10$ and LUT size $k = 4$. The input and output pins of a CLB can be connected to the wire segments in the surrounding channels via a connection

[2]However, the emphasis in [15] is architecture evaluation considering Vdd programmability.

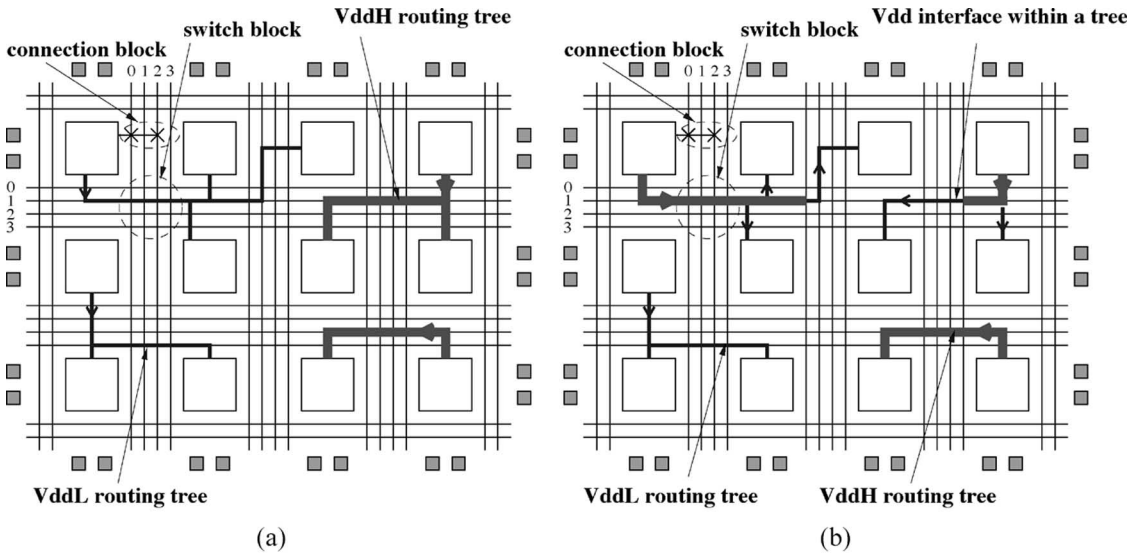


Fig. 1. (a) TLC. (b) dTLC.

block. There is a routing switch block at each intersection of a horizontal channel and a vertical channel. An interconnect switch is either a routing switch or a connection switch. For the rest of this paper, we use switch to represent interconnect switch for simplicity whenever there is no ambiguity. As suggested in [22], we assume a uniform length of 4, which is the optimal single wire length, for all wire segments. We plan to extend our methodology to the mix of different wire segment lengths in the future.

Vdd programmability can be applied to interconnects to reduce FPGA power. We use the Vdd-programmable interconnect switch with the minimal number of configuration SRAM cells proposed in [23]. Two PMOS power transistors M3 and M4 are inserted between the tristate buffer and VddH, VddL power rails, respectively. Turning off one of the power transistors we can select a Vdd level for the routing switch. By turning off both power transistors, an unused routing switch can be power gated. SPICE simulation shows that power gating the routing switch can reduce the leakage power of an unused routing switch by a factor of over 300 [23]. There are power and delay overheads associated with the power transistors insertion. The dynamic power overhead is almost negligible. This is because power transistors stay either ON or OFF after configuration, and there is no charging and discharging at their source/drain capacitors. The delay overhead associated with the power transistor insertion can be bounded when the power transistor is properly sized. Similar to the routing switch, a programmable Vdd is also applied to the connection switch. As in [13], we start with the single Vdd placed and routed netlists for MCNC benchmark circuits and then perform Vdd-level assignment for interconnects. To make the presentation simple, we summarize the notations frequently used in this paper in Table II. They will be explained in detail when first used.

TABLE II
NOTATIONS FREQUENTLY USED IN THIS PAPER

$\mathcal{G}(\mathcal{V},\mathcal{E})$	timing graph
$\mathcal{PI}$	set of all primary inputs and register outputs
$\mathcal{PO}$	set of all primary outputs and register inputs
$\mathcal{FO}_v$	set of all fanout vertices of vertex v in $\mathcal{G}$
$\mathcal{SRC}$	set of vertices corresponding to routing tree sources
$\mathcal{R}_i$	i^{th} routing tree in FPGA
$\mathcal{FO}_{ij}$	set of fanout switches of j^{th} switch in $\mathcal{R}_i$
$\mathcal{SL}_{ij}$	set of sinks in the fanout cone of j^{th} switch in $\mathcal{R}_i$
$a(v)$	arrival time of vertex v in $\mathcal{G}$
$d(u,v)$	delay from vertex u to vertex v in $\mathcal{G}$
N_r	total number (#) of routing trees in FPGA
v_{ij}	Vdd-level of j^{th} switch in $\mathcal{R}_i$
l_{ik}	# of switches in the path from source to k^{th} sink in $\mathcal{R}_i$
s_{ik}	allocated slack for k^{th} sink in $\mathcal{R}_i$
p_{i0}	vertex in $\mathcal{G}$ corresponding to the source of $\mathcal{R}_i$
p_{ik}	vertex in $\mathcal{G}$ corresponding to k^{th} sink of $\mathcal{R}_i$
$f_s(i)$	transition density of $\mathcal{R}_i$
$N_k(i)$	# of sinks in $\mathcal{R}_i$
$N_s(i)$	total # of switches in $\mathcal{R}_i$
$N_l(i)$	# of VddL switches in $\mathcal{R}_i$
$F_n(i)$	estimated # of VddL switches in $\mathcal{R}_i$

B. Delay Modeling With Dual Vdd

A directed acyclic timing graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ [22] is constructed to model the circuit for timing analysis. Vertices represent the input and output pins of basic circuit elements such as registers and LUTs. Edges are added between the inputs of combinational logic elements (e.g., LUTs) and their outputs, and between the connected pins specified by the circuit netlist. Register inputs are not joined to register outputs. Each edge is annotated with the delay required to pass through the circuit element or routing. We use $\mathcal{PI}$ to represent the set of primary inputs and register outputs that have no incoming edges, and $\mathcal{PO}$ to represent the set of primary outputs and register inputs that have no outgoing edges.

An Elmore delay model [24] is used to calculate the routing delay. We define the fanout cone of an interconnect switch as the subtree of the routing tree rooted at the switch. Assigning VddL to a switch affects the delay from the source to all the sinks in its fanout cone and therefore affects the delay of the corresponding edges in $\mathcal{G}$. To incorporate dual Vdd into the timing analysis, we use SPICE to precharacterize the intrinsic delay and effective driving resistance for a switch under VddH

TABLE III
CALIBRATED INPUT AND LOAD CAPACITANCE OF 7× BUFFER USING SPICE UNDER 1.3 V AND 0.8 V, RESPECTIVELY

Vdd (volt)	C_{input} (fF)	C_{load} (fF)
1.3	2.3	9.8
0.8	2.2	10.2

and VddL, respectively. We use the Berkeley predictive device model [25] at the ITRS 100-nm technology node in this paper. SPICE simulation (see Table III) shows that Vdd level has little impact on the input and load capacitance of a switch, and such impact is ignored in this paper.

C. Power Modeling With Dual Vdd

There are three power sources in FPGAs, namely, 1) switching power, 2) short-circuit power, and 3) leakage power. The first two contribute to the dynamic power and can only occur when a signal transition happens at the gate output. Although timing change may affect the transition density, we assume that the transition density for an interconnect switch will not change when VddL is used and the switches within one routing tree have the same transition density. The third type of power, leakage power, is the power consumed when there is no signal transition for a circuit element. We assume that the unused switches are power gated and consume no leakage. Despite simplification in the modeling, a more accurate power simulation will be performed to verify the experimental results in Section IV.

Given the Vdd level of interconnect switches and transition density of routing trees, the interconnect power P using a programmable dual Vdd can be expressed as the sum of dynamic power and leakage power as

$$P = 0.5 f_{\text{clk}} c \sum_{i=0}^{N_r-1} f_s(i) \sum_{j=0}^{N_s(i)-1} \text{Vdd}_{ij}^2 + \sum_{i=0}^{N_r-1} \sum_{j=0}^{N_s(i)-1} P_s(\text{Vdd}_{ij}) \quad (1)$$

where N_r is the total number of routing trees, $f_s(i)$ is the transition density of the ith routing tree $\mathcal{R}_i$, $N_s(i)$ is the number of switches in $\mathcal{R}_i$, and Vdd_{ij}, $P_s(\text{Vdd}_{ij})$, and c are the Vdd level, leakage power, and load capacitance of each switch, respectively. For the rest of this paper, we use $\mathcal{R}_i$ to represent the ith routing tree. Dynamic power quadratically depends on the Vdd level while leakage power exponentially depends on the Vdd level. For simplicity, we assume that all the switches have the same load capacitance. Our algorithms can, however, be extended to remove the simplification. v_{ij} indicates the Vdd level of the jth switch in $\mathcal{R}_i$ as

$$v_{ij} = \begin{cases} 1, & \text{if Vdd level of } j\text{th switch in } \mathcal{R}_i \text{ is VddH} \\ 0, & \text{if Vdd level of } j\text{th switch in } \mathcal{R}_i \text{ is VddL} \end{cases}.$$

Reducing the Vdd level can reduce both dynamic and leakage power. Interconnect power reduction P_r using a programmable dual Vdd can be expressed as the sum of dynamic power reduction and leakage power reduction as

$$\begin{aligned} P_r &= 0.5 f_{\text{clk}} c \Delta\text{Vdd}^2 \sum_{i=0}^{N_r-1} f_s(i) N_l(i) + \sum_{i=0}^{N_r-1} \Delta P_s N_l(i) \\ &= \sum_{i=0}^{N_r-1} \left[0.5 f_{\text{clk}} c \Delta\text{Vdd}^2 f_s(i) + \Delta P_s \right] N_l(i) \\ \Delta\text{Vdd}^2 &= \text{VddH}^2 - \text{VddL}^2 \\ N_l(i) &= \sum_{j=0}^{N_s(i)-1} (1 - v_{ij}) \end{aligned} \quad (2)$$

where $N_l(i)$ is the number of VddL switches that can be achieved in $\mathcal{R}_i$ and ΔP_s is the leakage power reduction of a switch by changing its supply voltage from VddH to VddL. We assume that unused switches have been power gated in this paper.

D. Problem Formulation

Removing Vdd-level converters requires that no VddL switch should drive VddH switches. For TLC, we enforce that there is only one Vdd level within each routing tree. Since two routing trees will not intersect with each other, we do not need level converters in routing channels. Fig. 1(a) shows the TLC and illustrates the situation that a VddH routing tree and VddL routing tree can share the same routing track without level converters in routing channels. The Vdd-level constraints for TLC can be expressed as

$$v_{ij} = v_{ik}, \qquad 0 \le i < N_r \wedge 0 \le j, \qquad k < N_s(i) \quad (3)$$

that is, each pair of switches within a routing tree has the same Vdd level. For dTLC, we can have different Vdd levels within one routing tree but no VddL switch drives VddH switches [see Fig. 1(b)]. The Vdd-level constraints for dTLC can be expressed as

$$v_{ik} \le v_{ij}, \qquad 0 \le i < N_r \wedge 0 \le j < N_s(i) \wedge k \in \mathcal{FO}_{ij} \quad (4)$$

that is, no VddL switch should drive VddH switches. $\mathcal{FO}_{ij}$ gives the set of fanout switches of the jth switch in $\mathcal{R}_i$.

The timing constraints require that the maximal arrival time at $\mathcal{PO}$ with respect to $\mathcal{PI}$ is at most T_{spec}, i.e., for all paths from $\mathcal{PI}$ to $\mathcal{PO}$, the sum of edge delays in each path p must be at most T_{spec}. As the number of paths from $\mathcal{PI}$ to $\mathcal{PO}$ can be exponential, the direct path-based formulation on timing constraints is impractical for analysis and optimization. Therefore, we use the net-based formulation that partitions the constraints on path delay into constraints on delay across circuit

Tree based algorithm:
Assign VddH to all routing trees and mark them as 'untried';
Calculate power-sensitivity for all routing trees;
While(∃ 'untried' routing tree)
{
Assign VddL to the routing tree with the largest power sensitivity if critical path increase does not increase;
Mark the routing tree as 'tried';
}

Fig. 2. Sensitivity-based algorithm (TLC-S) for TLC problem.

elements or routing. Let $a(v)$ be the arrival time for vertex v in $\mathcal{G}$ and the timing constraints become

$$a(v) \le T_{\text{spec}} \qquad \forall v \in \mathcal{PO} \tag{5}$$

$$a(v) = 0 \qquad \forall v \in \mathcal{PI} \tag{6}$$

$$a(u) + d(u,v) \le a(v) \qquad \forall u \in \mathcal{V} \wedge v \in \mathcal{FO}_u \tag{7}$$

where $\mathcal{V}$ is the set of vertices in $\mathcal{G}$, $d(u,v)$ is the delay from vertex u to v, and $\mathcal{FO}_u$ is the set of fanout vertices of u.

The objective function

$$\text{Maximize} \sum_{i=0}^{N_r-1} \left[0.5 f_{\text{clk}} c\, \Delta\text{Vdd}^2 f_s(i) + \Delta P_s\right] N_l(i) \tag{8}$$

is to maximize the power reduction (2). The TLC problem consists of objective function (8), Vdd-level constraints (3), and timing constraints (5)–(7). The dTLC problem is same as the tree-based problem except that Vdd-level constraints (4) replace (3).

III. Chip-Level VDD Assignment

A. TLC

In this section, we present a simple yet practical power sensitivity-based algorithm, namely, TLC-S, for TLC problem. Starting with a placed and routed single-Vdd circuit netlist, we calculate power sensitivity $\Delta P/\Delta\text{Vdd}$, which is defined as the power reduction (ΔP) divided by the Vdd-level difference (ΔVdd) between VddH and VddL, for each switch with the wire it drives. Total power P includes both the dynamic power P_d and the leakage power P_s. We define the power sensitivity of tree $\mathcal{R}_i$ as $\sum_{j=0}^{N_s(i)-1} \Delta P_{ij}/\Delta\text{Vdd}$, where $\Delta P_{ij}/\Delta\text{Vdd}$ is the power sensitivity of the jth switch in $\mathcal{R}_i$.

A greedy algorithm similar to that in [26] is performed to assign a Vdd level for routing trees (see Fig. 2). In the beginning, VddH is assigned to all the routing trees and the power sensitivity is calculated for each routing tree. We then iteratively perform the following steps. VddL is assigned to the routing tree with the largest power sensitivity. After updating the circuit timing, we accept the assignment if the critical path delay does not increase. Otherwise, we reject the assignment and restore the Vdd level of this routing tree to VddH. In either case, the routing tree will be marked as "tried" and will not be revisited in subsequent iterations. After the dual-Vdd assignment, we obtain a dual-Vdd netlist without performance loss.

Segment based algorithm:
Assign VddH to all switches and mark them as 'untried';
Calculate power-sensitivity for all switches;
While(∃ 'untried' switch)
{
Assign VddL to the candidate switch j with the largest power sensitivity;
If (critical path delay increases)
{
Find all the upstream switches of j in the same tree;
Assign VddH to j and those upstream switches, and mark them as 'tried';
}
Else mark j as 'tried';
}

Fig. 3. Sensitivity-based algorithm (dTLC-S) for dTLC problem.

B. dTLC

In this section, we present two Vdd-level assignment algorithms for the dTLC problem. The first algorithm is a sensitivity-based algorithm called dTLC-S, which is similar to the TLC-S presented above. Both TLC-S and dTLC-S implicitly allocate time slack first to routing trees or switches with a higher power sensitivity to reduce more power. An LP-based algorithm with explicit time slack allocation, namely, dTLC-LP is then presented. The details of these two algorithms are presented.

1) Sensitivity Based Algorithm (dTLC-S): The sensitivity-based algorithm dTLC-S is quite similar to TLC-S except for two differences. First, the assignment unit in dTLC-S is an interconnect switch instead of a routing tree. A switch is defined as a candidate switch if it follows two categories. In the first category, a candidate switch is "untried" and does not drive any switch. In the second category, VddL has been assigned to all the fanout switches of a candidate switch. In the assignment, we try to assign VddL to the candidate switch with maximum power sensitivity in each iteration. Second, when VddL cannot be assigned to a candidate switch due to timing constraints, we mark all the upstream switches of that candidate switch in the same routing tree as "tried" and those upstream switches stay VddH. As there is no level converter in routing channels, VddH has to be assigned to all the upstream switches of a VddH switch within a routing tree. There is no performance loss in dTLC-S, which is summarized in Fig. 3.

2) LP-Based Algorithm (dTLC-LP): The sensitivity-based algorithms TLC-S and dTLC-S implicitly allocate time slack first to routing trees or switches with higher power sensitivity to reduce more power. We present an LP-based algorithm, called as dTLC-LP with explicit time slack allocation considering both global and local optimality. As dTLC, in general, reduces more power than TLC, we only consider the LP-based algorithm for the dTLC problem. dTLC-LP includes three phases. We first allocate time slack to each routing tree by formulating the problem as an LP problem to maximize a lower bound of power reduction. We then perform a bottom-up assignment algorithm to achieve the optimal solution within each routing tree given the allocated time slack. We finally perform a

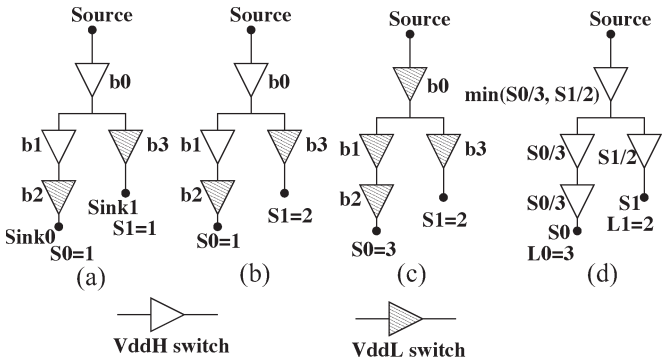


Fig. 4. Example for estimating number of VddL switches.

refinement to leverage surplus time slack. The details are discussed as follows.

1) Chip-level Time Slack Allocation

Estimation for number of low-Vdd switches: The slack s_{ij} of a connection between the source and the jth sink in $\mathcal{R}_i$ is defined as the amount of delay which could be added to this connection without increasing the cycle time T_{spec}. We represent the slack s_{ij} as a multiple of Δd, where Δd is the delay increase for an interconnect segment by changing the Vdd level from VddH to VddL. Fig. 4 presents a two-sink routing tree as an example. $S0$ and $S1$ are the slacks allocated to $Sink0$ and $Sink1$, respectively. In Fig. 4(a), VddL can be assigned to $b2$ given $S0 = 1$, and VddL can be assigned to $b3$ given $S1 = 1$. When we increase the slack $S1$ for $Sink1$ to 2 in Fig. 4(b), $b0$ has to stay VddH restricted by $S0 = 1$. In other words, $b0$ is restricted by both $S0$ and $S1$, and VddL can only be assigned to $b0$ when $S0 \geq 3 \wedge S1 \geq 2$. Fig. 4(c) shows the case in which VddL is assigned to all the switches given $S0 = 3 \wedge S1 = 2$. Therefore, there is an upper bound for slack, which is the delay increase when VddL is assigned to all the switches in a tree. Clearly, a slack more than the upper bound cannot lead to more VddL switches. We define the useful slack of each routing tree sink as the slack less than this upper bound. For the rest of this paper, we use slack to represent the useful slack.

Fig. 4(a) and (b) shows that we may achieve the same number of VddL switches with different slacks. Given a routing tree with an arbitrary topology and allocated slack for each sink, we need to estimate the number of VddL switches that can be achieved. We use l_{ik} to represent the number of switches in the path from the source to the kth sink in $\mathcal{R}_i$. We define sink list $\mathcal{SL}_{ij}$ as the set of sinks in the fanout cone of the jth switch in $\mathcal{R}_i$. We then estimate the number of VddL switches given the allocated slack as

$$F_n(i) = \sum_{j=0}^{N_s(i)-1} \min\left(\frac{s_{ik}}{l_{ik}} : \forall k \in \mathcal{SL}_{ij}\right). \tag{9}$$

To estimate the number of VddL switches in tree $\mathcal{R}_i$, we first deliberately distribute the slack s_{ik} evenly to l_{ik} switches in the path from the source to the kth sink in $\mathcal{R}_i$. For a switch with multiple sinks in its fanout cone, we choose the minimum s_{ik}/l_{ik} as the slack distributed to the switch. We then add the slack distributed to all the switches in $\mathcal{R}_i$ and get the estimated number of VddL switches. The rationale is that we consider the kth sink with minimum s_{ik}/l_{ik} in sink list $\mathcal{SL}_{ij}$ as the most critical sink to the jth switch in $\mathcal{R}_i$. Fig. 4(d) gives an example, and the estimated number of VddL switches is calculated as

$$F_n = \frac{S0}{3} + \frac{S0}{3} + \frac{S1}{2} + \min\left(\frac{S0}{3}, \frac{S1}{2}\right).$$

Theorem 1: Given a routing tree and allocated slack as a multiple of Δd, (9) gives a lower bound of the number of VddL interconnect switches that can be achieved.

It is easy to see that (9) gives the exact number of VddL switches for a one-sink tree. For a two-sink tree, we can verify that (9) gives a lower bound of the number of VddL switches with different allocated slack for each sink. Suppose this proposal of lower bound holds for any tree with $n-1$ sinks, we can prove that it is true for any n-sink routing tree. The details of proof are included in the technical report [27].

LP problem formulation: The objective function (8) is to maximize the power reduction, which is the weighted sum of the VddL switch number within each routing tree. To incorporate (9), which gives a lower bound of VddL switch number, into mathematical programming, we introduce a variable $f_n(i,j)$ for the jth switch in $\mathcal{R}_i$ and some additional constraints. The new objective function after transformation plus the additional constraints can be expressed as

$$\text{Maximize} \sum_{i=0}^{N_r-1} \left[0.5 f_{\text{clk}} c\, \Delta\text{Vdd}^2 f_s(i) + \Delta P_s\right] F_n(i) \tag{10}$$

s.t.

$$F_n(i) = \sum_{j=0}^{N_s(i)-1} f_n(i,j) 0 \leq i < N_r \wedge 0 \leq j < N_s(i) \tag{11}$$

$$f_n(i,j) \leq \frac{\lfloor s_{ik} \rfloor}{l_{ik}} \quad 0 \leq i < N_r \wedge 0 \leq j < N_s(i) \wedge \forall k \in \mathcal{SL}_{ij}. \tag{12}$$

Slack s_{ik} is a continuous variable normalized to Δd in (12) and also the rest of this paper. The floor function $\lfloor s_{ik} \rfloor$ in (12) gives a multiple of Δd and is introduced to make (11) a lower bound of the number of VddL switches. To avoid the floor function that is not a linear operation, we replace $\lfloor s_{ik} \rfloor / l_{ik}$ with $(s_{ik} - 1)/l_{ik}$ in (12) and have

$$f_n(i,j) \leq \frac{s_{ik} - 1}{l_{ik}} 0 \leq i < N_r \wedge \forall k \in \mathcal{SL}_{ij}. \tag{13}$$

The slack upper bound constraints can be expressed as

$$0 \leq s_{ik} \leq l_{ik}, \qquad 0 \leq i < N_r \wedge 1 \leq k \leq N_k(i) \tag{14}$$

where $N_k(i)$ is the number of sinks in $\mathcal{R}_i$.

We modify the timing constraints (7) as follows. For the edges corresponding to routing in $\mathcal{G}$, the constraints considering slack can be expressed as

$$a(p_{i0}) + d(p_{i0}, p_{ik}) + s_{ik}\Delta d \leq a(p_{ik}) \quad 0 \leq i < N_r \wedge \forall p_{ik} \in \mathcal{FO}_{p_{i0}} \tag{15}$$

```
Bottom-up assignment within R_i:
Assign VddH to all switches in R_i and mark them as 'untried';
While( ∃ candidate switch)
{
    Assign VddL to a candidate switch j;
    If (timing constraints violated)
    {
        Find all the upstream switches of j in R_i;
        Assign VddH to j and those upstream switches, and
mark them as 'tried';
    }
    Else mark j as 'tried';
}
```

Fig. 5. Net-level bottom-up assignment.

where vertex p_{i0} is the source of $\mathcal{R}_i$ in $\mathcal{G}$, vertex p_{ik} is the kth sink of $\mathcal{R}_i$ in $\mathcal{G}$, and $d(p_{i0}, p_{ik})$ is the delay from p_{i0} to p_{ik} in $\mathcal{R}_i$ using VddH. For edges other than routing in $\mathcal{G}$, the constraints can be expressed as

$$a(u) + d(u, v) \leq a(v) \qquad \forall u \in \mathcal{V} \wedge u \notin \mathcal{SRC} \wedge v \in \mathcal{FO}_u \tag{16}$$

where $\mathcal{SRC}$ is a subset of $\mathcal{V}$ and gives the set of vertices corresponding to routing tree sources.

We formulate the time slack allocation problem using objective function (10), additional constraints (11) and (13), slack upper bound constraints (14), plus timing constraints (5), (6), (15), and (16). It is easy to verify that (5), (6), (11), and (13)–(16) are linear, and the objective function (10) is linear too. Hence, we have the following theorem.

Theorem 2: The time slack allocation problem is an LP problem.

There are well-developed LP solvers available from both the commercial world like [28] and the academia like [29]. In this paper, we use the LP solver from [29]. For the rest of this paper, we use the LP problem to represent the time slack allocation problem.

2) Net-Level Assignment

Given the allocated slack after solving the LP problem, we perform a bottom-up assignment within each tree to leverage the allocated slack (see Fig. 5). For each tree $\mathcal{R}_i$, VddH is first assigned to all the switches in $\mathcal{R}_i$. We then iteratively perform the following steps in a bottom-up fashion. We assign VddL to a candidate switch and mark the switch as "tried." After updating the circuit timing, we reject the assignment and restore the Vdd level of the switch to VddH if the delay increase at any sink exceeds the allocated slack. The iteration terminates when there is no candidate switch in $\mathcal{R}_i$.

Theorem 3: Given a routing tree $\mathcal{R}_i$ and allocated slack for each sink, the bottom-up assignment gives the optimal assignment solution when Vdd-level converters cannot be used.

Sketch of proof: If the jth switch in $\mathcal{R}_i$ is assigned to VddL in the solution given by the bottom-up assignment and is assigned to VddH in an optimal solution, we can assign VddL to the jth switch and all of its downstream switches in the optimal solution without violating timing constraints and get another solution better than the optimal solution. Therefore, the bottom-up assignment algorithm gives the optimal solution when level converters cannot be used. ■

Theorem 4: Given a routing tree $\mathcal{R}_i$ in which each switch has a uniform load capacitance and transition density of the routing tree,[3] and a Vdd-level converter can be used, there exists a power-optimal Vdd-level assignment for any given slack without using Vdd-level converters.

Sketch of proof: In an optimal solution using level converters, each VddL switch in $\mathcal{R}_i$ can drive at most one VddH switch; otherwise, we can swap the Vdd level of the VddL switch and its fanout VddH switches without violating timing constraints and achieves more VddL switches than the optimal solution. Given this observation, for each VddL switch driving one VddH switch in an optimal solution, we can swap the Vdd level of the VddL switch and its fanout VddH switch without introducing more level converters. By keeping this process, we can eventually achieve a solution with the same number of VddH and VddL switches as the optimal solution, but no level converter is needed. ■

3) Refinement

After net-level assignment, we may further reduce power by leveraging surplus slack. Fig. 4(b) shows a routing tree containing surplus slack. $b0$ has to stay VddH due to the fact that it is restricted by $S0 = 1$. Therefore, $Sink1$ can only consume one unit slack from $S1$ and there is a surplus slack of 1. To leverage surplus slack, we mark all the VddH switches as "untried" but keep the VddL switches as "tried," and then perform the sensitivity-based algorithm dTLC-S (see Fig. 3) to achieve more VddL switches and further reduce power.

IV. Experimental Results

In this section, we conduct experiments on the MCNC benchmark set. We first compare the interconnect power and runtime between sensitivity-based algorithms (TLC-S and dTLC-S) and LP-based algorithm (dTLC-LP). We then compare the best one among TLC-S, dTLC-S, and dTLC-LP, and one previous approach without using level converters in interconnects (h2lLCi [12]) to the baseline using Vdd-programmable interconnects with SLC [3]. We use the same Vdd-programmable logic blocks and interconnects in [23] with no level converter inserted in routing channels. The unused interconnect switches are power gated in all cases. Same as [13], we customize the FPGA chip size for each benchmark circuit and use the smallest chip that just fits each benchmark. Considering the VddL/VddH ratio between 0.6 and 0.7 suggested in [30], we use 1.3v for VddH and 0.8v for VddL in our experiments at 100-nm technology node.

We first use VPR [22] for single Vdd placement and routing. Before applying SLC, TLC-S, dTLC-S, or dTLC-LP to Vdd-programmable interconnects, a sensitivity-based assignment [10] is first performed to assign a Vdd level for Vdd-programmable logic blocks without performance loss.[4] It is

[3]For a buffered interconnect tree, all the buffers in the tree have the same transition density without considering glitches, which might be weakened or absorbed after propagating through a few buffers. Nevertheless, our problem formulations and algorithms can be extended if the transition density is known for each individual buffer.

[4]All the algorithms SLC, TLC-S, dTLC-S, and dTLC-LP do consider the fact that VddL logic blocks consume time slack.

TABLE IV
PERCENTAGE OF VddL SWITCHES AND INTERCONNECT POWER ACHIEVED BY TLC-S, dTLC-S, AND dTLC-LP

1	2	3	4	5	6	7	8	9	10	11	12	13
circuit	% of VddL switches			interconnect power			interconnect dynamic power			interconnect leakage power		
	TLC-S	dTLC-S	dTLC-LP (due to refinement)	TLC-S (watt)	dTLC-S	dTLC-LP (due to refinement)	TLC-S (watt)	dTLC-S	dTLC-LP (due to refinement)	TLC-S (watt)	dTLC-S	dTLC-LP (due to refinement)
alu4	33.52%	58.72%	60.36% (3.04%)	0.05225	-12.49%	-14.70% (-1.07%)	0.04948	-12.75%	-15.06% (-1.08%)	0.00277	-7.73%	-8.25% (-0.92%)
apex2	39.74%	68.94%	69.69% (4.53%)	0.07022	-20.42%	-24.01% (-2.46%)	0.06572	-21.13%	-24.95% (-2.53%)	0.00450	-10.03%	-10.29% (-1.56%)
apex4	31.97%	58.99%	58.25% (5.75%)	0.03340	-16.62%	-21.00% (-1.75%)	0.03027	-17.42%	-22.27% (-1.74%)	0.00312	-8.85%	-8.61% (-1.87%)
bigkey	65.65%	76.83%	77.44% (2.01%)	0.09110	-9.52%	-9.14% (-1.23%)	0.08601	-9.93%	-9.52% (-1.28%)	0.00509	-2.55%	-2.70% (-0.44%)
clma	56.58%	79.17%	80.79% (4.46%)	0.16407	-20.82%	-26.29% (-2.09%)	0.13008	-24.22%	-30.97% (-2.24%)	0.03399	-7.82%	-8.38% (-1.51%)
des	60.03%	79.90%	80.38% (2.68%)	0.10906	-11.40%	-13.44% (-1.83%)	0.10266	-11.82%	-13.97% (-1.91%)	0.00641	-4.80%	-4.97% (-0.57%)
diffeq	80.52%	91.23%	91.56% (3.22%)	0.00771	-2.47%	-2.17% (-4.72%)	0.00467	-1.85%	-1.28% (-7.14%)	0.00304	-3.43%	-3.54% (-1.00%)
dsip	61.01%	77.69%	78.01% (1.28%)	0.10546	-12.80%	-12.91% (-1.09%)	0.10023	-13.29%	-13.41% (-1.13%)	0.00524	-3.36%	-3.43% (-0.27%)
elliptic	79.46%	93.69%	94.09% (1.51%)	0.02483	-3.01%	-3.46% (-1.32%)	0.01685	-2.05%	-2.64% (-1.70%)	0.00798	-5.05%	-5.19% (-0.53%)
ex1010	43.93%	69.21%	69.66% (5.99%)	0.06501	-20.05%	-22.93% (-1.37%)	0.05336	-22.59%	-26.06% (-1.23%)	0.01164	-8.45%	-8.60% (-2.00%)
ex5p	38.12%	65.50%	64.45% (4.72%)	0.02818	-14.68%	-15.09% (-1.29%)	0.02512	-15.40%	-15.89% (-1.27%)	0.00306	-8.84%	-8.49% (-1.48%)
frisc	95.80%	99.06%	99.07% (8.99%)	0.02431	-1.25%	-1.34% (-5.00%)	0.01167	-1.43%	-1.61% (-7.16%)	0.01264	-1.08%	-1.08% (-3.00%)
misex3	37.12%	64.58%	65.90% (3.26%)	0.05168	-14.90%	-18.38% (-1.56%)	0.04869	-15.25%	-18.92% (-1.59%)	0.00298	-9.10%	-9.56% (-1.08%)
pdc	37.65%	71.34%	72.24% (3.79%)	0.09887	-23.32%	-26.44% (-1.74%)	0.08346	-25.38%	-29.01% (-1.82%)	0.01541	-12.17%	-12.50% (-1.35%)
s298	39.18%	81.28%	81.88% (2.00%)	0.02445	-28.76%	-31.04% (-0.39%)	0.02090	-31.21%	-33.84% (-0.34%)	0.00355	-14.30%	-14.50% (-0.68%)
s38417	75.16%	85.40%	86.31% (3.47%)	0.09803	-6.58%	-9.88% (-3.43%)	0.08295	-7.20%	-11.05% (-3.86%)	0.01508	-3.15%	-3.43% (-1.04%)
s38584	87.56%	94.42%	94.56% (3.57%)	0.08829	-5.49%	-6.74% (-1.56%)	0.07637	-6.00%	-7.44% (-1.65%)	0.01192	-2.20%	-2.26% (-1.02%)
seq	33.04%	61.38%	62.21% (3.38%)	0.07198	-18.36%	-22.21% (-2.20%)	0.06765	-18.95%	-23.03% (-2.28%)	0.00434	-9.13%	-9.43% (-1.07%)
spla	32.08%	69.35%	70.69% (4.02%)	0.07343	-24.89%	-27.87% (-2.24%)	0.06377	-26.72%	-30.08% (-2.37%)	0.00966	-12.79%	-13.26% (-1.34%)
tseng	93.34%	95.88%	96.55% (3.33%)	0.00850	-1.27%	0.22% (-6.49%)	0.00643	-1.40%	0.64% (-8.26%)	0.00207	-0.86%	-1.08% (-1.00%)
avg.	56.07%	77.13%	77.70% (3.75%)	-	-13.45%	-15.44% (-2.24%)	-	-14.30%	-16.52% (-2.63%)	-	-6.79%	-6.98% (-1.19%)

easy to see that our algorithms, especially sensitivity-based algorithms, can be easily extended to consider Vdd assignment for both logic blocks and interconnects in a uniform fashion. The cycle-accurate FPGA power simulator fpgaEva-LP2 [15] is then used to calculate power. It has been shown that fpgaEva-LP2 achieves high fidelity as well as high accuracy as compared to SPICE simulation with an average of absolute error 8.26% [15]. Because the power computation in fpgaEva-LP2 considers short circuit power and uses input vectors, it is more accurate than the power model in our problem formulations. Using fpgaEva-LP2 verifies both our modeling and problem formulations.

A. Comparison Between TLC-S, dTLC-S, and dTLC-LP

1) Interconnect Power Comparison: We present the number of VddL switches and interconnect power achieved by TLC-S, dTLC-S, and dTLC-LP in Table IV. The number of VddL switches is expressed in percent of used switch number. The interconnect power achieved by dTLC-S and dTLC-LP is presented in the power difference normalized to TLC-S in this section unless specified otherwise. Columns 2–4 in Table IV present the percentage of VddL switches achieved by the three algorithms. TLC-S, dTLC-S, and dTLC-LP achieve 56.07%, 77.13%, and 77.70% VddL switches, respectively. dTLC-S and dTLC-LP achieve almost the same VddL switches. Both of these two algorithms achieve more VddL switches than TLC-S. This is because TLC uses a routing tree as the assignment unit and does not allow the mix of different Vdd levels within a routing tree.

Columns 5–7 in Table IV present the overall interconnect power achieved by the three algorithms. Compared to TLC-S, dTLC-S and dTLC-LP consume 13.45% and 15.44% less power, respectively. We also present the interconnect dynamic power and leakage power in columns 8–13. Compared to TLC-S, dTLC-S and dTLC-LP consume 14.30% and 16.52% less dynamic power, 6.79% and 6.98% less leakage power, respectively. Clearly, dTLC-LP achieves the lowest interconnect power consumption. This is because dTLC-LP considers both global and local optimality. Fig. 6 compares the estimated

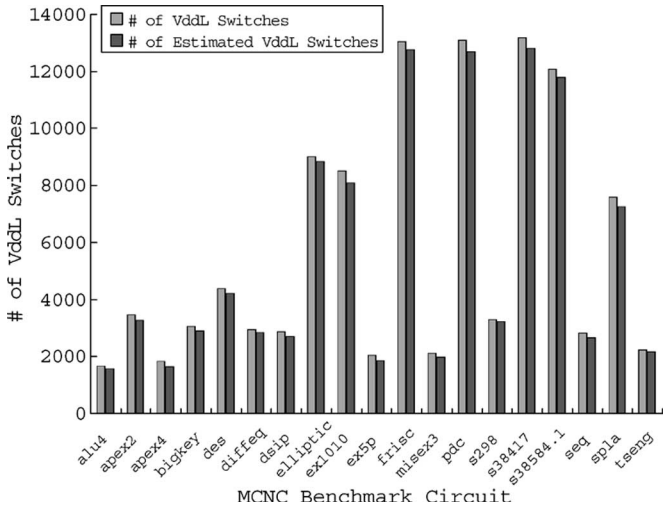


Fig. 6. Comparison between VddL switch number achieved by dTLC-LP before refinement and estimated VddL switch number.

VddL switch number given in (11) and the number of VddL switches given by dTLC-LP without refinement, i.e., the number of VddL switches achieved by the bottom-up assignment given the allocated slack. The comparison shows that (11) has an average error of 4.68%, and it has both high fidelity and accuracy.

For dTLC-LP, we also present the contribution of refinement step in columns 4, 7, 10, and 13 in Table IV. The refinement step achieves 3.75% VddL switches. Compared to TLC-S, the refinement step obtains 2.24% interconnect power reduction, 2.63% interconnect dynamic power reduction, and 1.19% interconnect leakage power reduction. It is clear that the refinement step is effective to redistribute surplus time slack and to further reduce interconnect power after chip-level time slack allocation and net-level bottom-up assignment. This existing surplus time slack is mainly due to two reasons. The first source is the difference between the continuous LP formulation (i.e., the allocated slack could be continuous) and the fact that Vdd-level assignment is discrete (i.e., the slack consumed by VddL switches must be a multiple of Δd). Second, the objective of the

TABLE V
RUNTIME COMPARISON BETWEEN TLC-S, dTLC-S, AND dTLC-LP

circuit	# of nodes	runtime (s)		
		TLC-S	dTLC-S	dTLC-LP
alu4	10716	60.52	124.4	482.53
apex2	14860	180.75	378.59	1153.28
apex4	9131	66.93	177.52	461.37
bigkey	18622	321.22	416.42	1343.65
clma	91620	8763.24	16799.67	55901.08
des	15243	176.65	287.81	1054.74
diffeq	13664	113.81	143.95	553.84
dsip	11444	96.45	131.62	406.96
elliptic	30192	607.85	913.04	3136.59
ex1010	33265	836.32	1422.79	5109.22
ex5p	8722	62.11	93.99	187.44
frisc	40662	1135.84	1912.15	6135.38
misex3	10271	74.35	106.72	276.41
pdc	40001	1254.57	2508.57	8210.07
s298	16852	179.72	238.18	837.22
s38417	57503	1821.09	2895.79	9152.52
s38584	46014	1255.31	1892.86	6863.62
seq	13426	129.22	203.01	509.22
spla	27908	524.76	1009.07	3339.51
tseng	9603	52.45	71.53	163.55
geometric mean		1*X*	1.64*X*	5.10*X*

LP formulation is to maximize a low bound of power reduction instead of the exact power reduction.

2) Runtime Comparison: Table V compares the runtime[5] between the three algorithms. TLC-S is the fastest among the three algorithms. dTLC-S and dTLC-LP take 1.64× and 5.10× runtime compared to the fastest one. Solving the LP problem contributes the largest part of the overall runtime of dTLC-LP. The refinement step in dTLC-LP takes less than 5% of the overall runtime. Note that MCNC benchmark circuits have already been partitioned into combinational circuit blocks. In general, large circuits might be partitioned and the LP problem might then be solved for each partition to reduce runtime. Compared to dTLC-LP, dTLC-S has a slightly larger power consumption, but runs three times faster and is effective for large circuits. dTLC-LP is worthwhile for small circuits and can achieve the best power reduction.

B. Justification of Vdd-Level Assignment Approach

In this paper, we first perform a sensitivity-based Vdd-level assignment for Vdd-programmable logic blocks before assigning a Vdd level for interconnects. Alternatively, we may allocate time slack first to interconnects and then to logic blocks (dTLC-S-I) or to logic blocks and interconnects in a uniform fashion (dTLC-S-U). Table VI compares the three algorithms, namely: 1) dTLC-S, 2) dTLC-S-I, and 3) dTLC-S-U, all implemented based on power sensitivity. Columns 2–7 in Table IV show that dTLC-S, dTLC-S-I, and dTLC-S-U achieve 74.17%, 24.20%, and 71.03% VddL CLBs, respectively. They achieve 77.13%, 85.75%, and 77.75% VddL switches, respectively. dTLC-S and dTLC-S-I achieve the highest percentage of VddL CLBs and interconnect switches, respectively. dTLC-S and dTLC-S-U achieve similar percentages of VddL CLBs and switches. Columns 8–10 in Table VI present the total power achieved by the three approaches. Compared to dTLC-S, dTLC-S-I and dTLC-S-U consume 13.20% and 0.39% more total power, respectively. It shows that allocating time slack first to CLBs is better than allocating slack first to interconnects. In addition, allocating slack first to CLBs is slightly better than (or as good as) allocating slack in a uniform fashion for CLBs and interconnects. It is because that CLBs tend to have larger power sensitivities than interconnect switches.

C. Comparison Between SLC, h2lLCi, and dTLC-LP With Relaxed Timing Specification

In this section, we compare dTLC-LP, which obtains the lowest interconnect power consumption among TLC-S, dTLC-S, and dTLC-LP, to two previous approaches SLC [13] and h2lLCi [12]. SLC inserts a level converter in front of each interconnect switch as well as each CLB input and output. A greedy sensitivity-based assignment is performed for the Vdd-programmable interconnects. Gayasen *et al.* [12] insert a level converter either at each CLB input or output. A path-based assignment is performed for Vdd-programmable interconnects. It has been shown that h2lLCi, which inserts a level converter at each CLB input and initializes all the circuit elements with VddH, achieves the lowest power consumption among all the proposed approaches in [12].

1) Interconnect Power Comparison With Relaxed Timing Specification: Timing specification may be relaxed for certain applications that are not timing critical. In this case, more VddL switches can be achieved and therefore more power can be reduced with relaxed timing specification. Fig. 7 compares the percentage of VddL switches and relaxed critical path delay tradeoff curves achieved by SLC, h2lLCi, and dTLC-LP. The VddL switch percentage and the critical path delay are the arithmetic and geometric means over the 20 largest MCNC benchmark circuits, respectively. When the timing specification is not relaxed, SLC, h2lLCi, and dTLC-LP achieve 74.79%, 41.98%, and 77.70%, respectively. Both SLC and dTLC-LP achieve more VddL switches than h2lLCi. It is because that all the trees driven by one CLB have the same Vdd level as the source CLB, and the VddH circuit element is not allowed to drive the VddL circuit element without the presence of a level converter in h2lLCi. dTLC-LP consistently achieves the highest percentage of VddL switches compared to previous approach SLC [6] and h2lLCi at different relaxed delays.

Fig. 8 compares the interconnect power and critical path delay tradeoff curves achieved by SLC, h2lLCi, and dTLC-LP. Both the interconnect power and the critical path delay are the geometric mean over the 20 largest MCNC benchmark circuits. Compared to SLC, h2lLCi and dTLC-LP reduce the interconnect power by 41.44% and 52.90% without relaxing timing specification, respectively. Compared to h2lLCi, dTLC-LP reduces interconnect power by 18.52% at the highest clock frequency. dTLC-LP consistently achieves the lowest power compared to SLC and h2lLCi at different relaxed delays. Compared

[5]The runtime includes single-Vdd placement and routing by VPR and generating the interface files between VPR and fpgaEva-LP2.

[6]Without considering the delay overhead of level converters, SLC [13] may achieve more VddL switches as Vdd-programmable interconnects with level converters are more flexible in Vdd-level assignment.

TABLE VI
PERCENTAGE OF VddL CLBs AND INTERCONNECT SWITCHES, AND TOTAL POWER ACHIEVED BY THREE APPROACHES OF dTLC-S

1	2	3	4	5	6	7	8	9	10
	% of VddL CLBs			% of VddL switches			total power		
Circuit	dTLC-S CLB → interconnects	dTLC-S-I interconnects → CLBs	dTLC-S-U uniform	dTLC-S CLB → interconnects	dTLC-S-I interconnects → CLBs	dTLC-S-U uniform	dTLC-S CLB → interconnects (watt)	dTLC-S-I interconnects → CLBs	dTLC-S-U uniform
alu4	54.94%	2.47%	54.94%	58.72%	70.93%	58.72%	0.07898	+14.63%	-0.35%
apex2	67.14%	0.94%	67.14%	68.94%	81.84%	68.94%	0.08462	+14.49%	-0.46%
apex4	51.49%	0.00%	49.25%	58.99%	68.09%	59.98%	0.04489	+13.38%	-0.47%
bigkey	81.29%	11.56%	81.29%	76.83%	88.58%	76.83%	0.15072	+28.27%	+0.66%
clma	84.24%	20.84%	81.08%	79.17%	88.40%	79.85%	0.19562	+12.33%	+0.44%
des	78.44%	16.51%	78.44%	79.90%	90.10%	79.90%	0.14224	+14.31%	-0.14%
diffeq	82.05%	53.33%	65.13%	91.23%	95.68%	93.71%	0.01652	+7.24%	+2.61%
dsip	56.79%	7.41%	56.79%	77.69%	87.84%	77.69%	0.15132	+10.38%	-0.04%
elliptic	93.82%	57.72%	89.07%	93.69%	97.71%	94.58%	0.04323	+12.07%	-0.19%
ex1010	78.90%	2.03%	71.81%	69.21%	81.40%	71.54%	0.07907	+17.66%	+0.55%
ex5p	36.59%	0.00%	34.96%	65.50%	74.28%	65.35%	0.04147	+8.79%	+0.35%
frisc	94.45%	85.88%	88.91%	99.06%	99.52%	99.26%	0.04535	+2.08%	+0.73%
misex3	71.24%	3.92%	71.24%	64.58%	79.40%	64.58%	0.06676	+16.18%	+0.30%
pdc	71.13%	3.35%	62.15%	71.34%	80.71%	73.51%	0.11355	+12.55%	-0.93%
s298	80.86%	29.30%	80.47%	81.28%	89.16%	81.52%	0.03270	+18.80%	-0.42%
s38417	85.71%	40.73%	84.30%	85.40%	93.60%	85.67%	0.15526	+15.26%	+1.45%
s38584	96.73%	67.61%	94.32%	94.42%	97.89%	94.97%	0.14219	+11.59%	+0.05%
seq	68.18%	2.02%	68.18%	61.38%	73.49%	61.38%	0.08841	+16.96%	+0.03%
spla	65.41%	2.76%	60.15%	69.35%	79.21%	70.36%	0.08392	+12.64%	+0.87%
tseng	83.97%	75.57%	80.92%	95.88%	97.17%	96.67%	0.01725	+4.47%	+2.68%
avg.	74.17%	24.20%	71.03%	77.13%	85.75%	77.75%	-	+13.20%	+0.39%

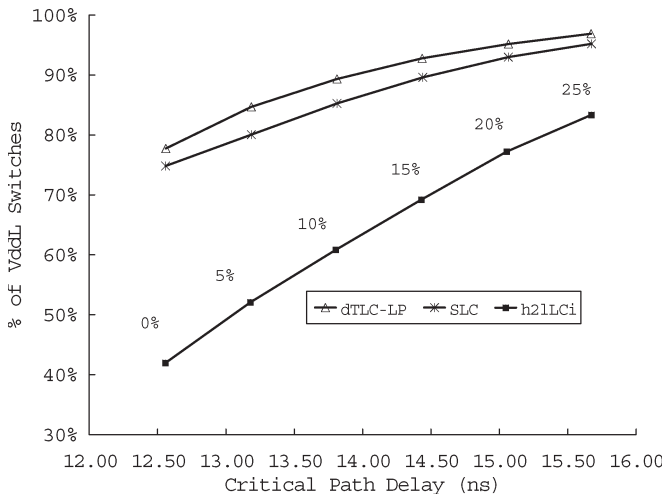


Fig. 7. Percentage of VddL interconnect switches versus critical path delay curves for SLC, h2lLCi, and dTLC-LP.

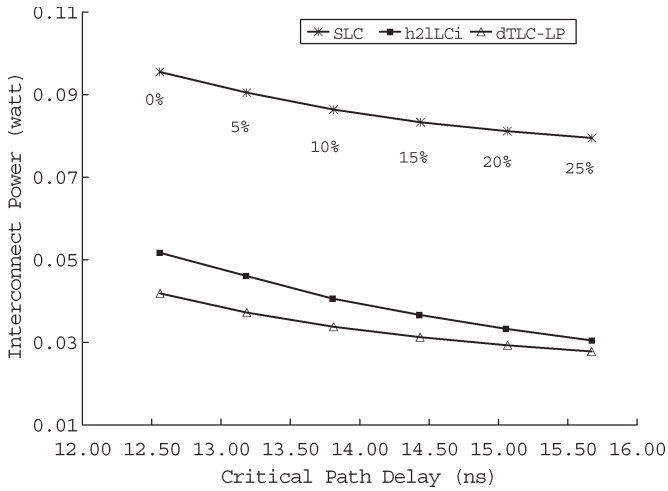


Fig. 8. Interconnect power and critical path delay tradeoff for SLC, h2lLCi, and dTLC-LP.

to SLC at the same relaxed delay, dTLC-LP achieves 89.31% VddL switches and reduces interconnect power by 58.18% when we relax the critical path delay by 10%. The power gap between dTLC-LP and h2lLCi decreases at a larger relaxed delay. This is because that VddL eventually can be assigned to all switches (see Fig. 7) and interconnect power reduction will saturate if we allow sufficient critical path increase.

2) Area Comparison: Table VII presents the FPGA total area given by a single Vdd, SLC, h2lLCi, and dTLC-LP, respectively. We use the same area model from [22], in which area is counted in number of minimum width transistor areas considering the parallel diffusions technique for large transistors. Given a transistor with channel width W, the transistor area measured by the minimum width transistor with channel width $W_{\min}$ is

$$\text{Area}(W) = 0.5 + \frac{W}{2W_{\min}}. \tag{17}$$

As presented in the table, the area overheads given by SLC, h2lLCi, and dTLC-LP are 151%, 65%, and 66% compared to the single-Vdd FPGA. h2lLCi and dTLC-LP have almost the same area, and TLC-S, dTLC-S, and dTLC-LP have the same area as all of them use the same Vdd-programmable interconnects with level converters inserted at CLB inputs and outputs.

V. CONCLUSION

To remove Vdd-level converters in routing channels that introduce large leakage and area overheads, we have proposed TLC, where there is only one Vdd level within each routing tree, and dTLC, where different Vdd levels can exist within a routing tree, but no VddL switch drives VddH switches. In both cases, configurable level converters are inserted at the inputs and outputs of logic blocks (CLBs) to allow the mix of Vdd levels for CLBs and interconnects.

TABLE VII
FPGA AREA COMPARISON BETWEEN A SINGLE Vdd, SLC, h2lLCi, AND dTLC-LP. AREA IS IN NUMBER OF MINIMUM WIDTH TRANSISTORS. WE USE 210× AND 4× PMOS POWER TRANSISTORS FOR CLBs AND 7× SWITCHES, RESPECTIVELY

circuit	single-Vdd area	SLC	h2lLCi	dTLC-LP
		area (overhead)	area (overhead)	area (overhead)
alu4	2925831	7136109 (144%)	4779206 (63%)	4795092 (64%)
apex2	4384857	11078412 (153%)	7215910 (65%)	7237060 (65%)
apex4	3007921	7534174 (150%)	4903820 (63%)	4917356 (63%)
bigkey	8614605	19628149 (128%)	14192729 (65%)	14261255 (66%)
clma	36361942	100634278 (177%)	61402657 (69%)	61531343 (69%)
des	12371141	28245400 (128%)	20338979 (64%)	20435235 (65%)
diffeq	3185418	7667824 (141%)	5199431 (63%)	5217855 (64%)
dsip	9718346	22792498 (135%)	15986128 (64%)	16054654 (65%)
elliptic	9056268	23394478 (158%)	15004948 (66%)	15046402 (66%)
ex1010	11605734	30506887 (163%)	19308966 (66%)	19358692 (67%)
ex5p	3106231	7808090 (151%)	5061245 (63%)	5074781 (63%)
frisc	17028484	46672107 (174%)	28575123 (68%)	28633873 (68%)
misex3	3063483	7548212 (146%)	5009146 (64%)	5025032 (64%)
pdc	15929127	43487207 (173%)	26652048 (67%)	26706192 (68%)
s298	3947796	9485182 (140%)	6470105 (64%)	6494169 (65%)
s38417	15975183	40827249 (156%)	26625521 (67%)	26710121 (67%)
s38584	13005036	33144130 (155%)	21637358 (66%)	21705884 (67%)
seq	4384857	11078412 (153%)	7215910 (65%)	7237060 (65%)
spla	9664508	25585907 (165%)	16038715 (66%)	16076315 (66%)
tseng	2275254	5397201 (137%)	3698578 (63%)	3712114 (63%)
avg.	-	- (151%)	- (65)%	- (66%)

We have developed a few Vdd-level assignment algorithms considering time slack allocation to maximize power reduction. Our Vdd-level assignment algorithms include two power sensitivity-based algorithms (TLC-S and dTLC-S) for the TLC and dTLC problems, respectively, and an LP-based algorithm (dTLC-LP) for the dTLC problem. TLC-S and dTLC-S implicitly allocate time slack first to interconnects with a higher power sensitivity and assign VddL to them for more power reduction. dTLC-LP first explicitly allocates time slack to each routing tree using LP, and then assigns a Vdd level within each routing tree for the allocated time slack. Experiments show that dTLC-LP obtains the lowest power consumption among all the formulations and algorithms. Compared to dTLC-LP, dTLC-S obtains a slightly higher power consumption but runs three times faster. Therefore, it is also an attractive algorithm with a desired tradeoff between runtime and quality. Without loss of performance, the best algorithm (dTLC-LP) reduces the total interconnect power by 52.90% compared to the SLC proposed in [13], and by 18.52% compared to the best approach h2lLCi proposed in [12], where all routing trees driven by one CLB have the same Vdd level as the source CLB.

State-of-the-art commercial FPGAs have used wire segments of different lengths and applied unidirectional level-restore routing switch in the routing architecture [31]–[33] to improve performance. The methodology proposed in this paper can be extended to interconnects with these novel features. We believe that dTLC formulation can still outperform TLC and h2lLCi as dTLC allows the mix of different Vdd levels within a routing tree and is more effective to leverage the time slack. The positive-feedback PMOS transistor in the level-restore buffer can be used as an alternative for a level converter [14]. This allows a VddL switch to drive VddH switches for potentially more dynamic power reduction, but introduces extra leakage, short-circuit power, and extra delay. Using level-restore buffers in the context of time slack allocation is planned as a future study. In addition, we plan to extend dTLC to consider wire segments of different lengths.

The algorithms proposed in this paper allocate time slack first to CLBs and then to interconnects. Using sensitivity-based algorithms, we have shown that this is better than allocating time slack first to interconnects and then to CLBs and is slightly better than (or as good as) allocating time slack in a uniform fashion to both interconnects and CLBs. In the future, such a comparison may be reconducted using an LP-based algorithm for the three ways of allocating time slack. In our paper, we perform Vdd-level assignment based on single-Vdd routing results, which may be suboptimal for Vdd-programmable interconnects. In the future, we will also study power-driven routing algorithms, which simultaneously perform routing and Vdd-level assignment for Vdd-programmable interconnects.

ACKNOWLEDGMENT

The authors would like to thank Dr. F. Li and J. Xiong of UCLA for helpful discussions. The authors used computers donated by Intel.

REFERENCES

[1] K. Poon, A. Yan, and S. Wilton, "A flexible power model for FPGAs," in *Proc. 12th Int. Conf. Field-Programmable Logic and Applications*, Montpellier, France, Sep. 2002, pp. 312–321.

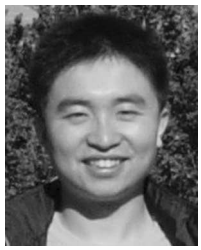
[2] F. Li, D. Chen, L. He, and J. Cong, "Architecture evaluation for power-efficient FPGAs," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, Monterey, CA, Feb. 2003, pp. 175–184.

[3] R. Mukherjee and S. O. Memik, "Power-driven design partitioning," in *Proc. Int. Conf. Field-Programmable Logic and Application*, Leuven, Belgium, Aug. 2004, pp. 740–750.

[4] J. Lamoureux and S. J. Wilton, "On the interaction between power-aware FPGA CAD algorithms," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 2003, pp. 701–708.

[5] J. H. Anderson, F. N. Najm, and T. Tuan, "Active leakage power optimization for FPGAs," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, Monterey, CA, Feb. 2004, pp. 33–41.

[6] S. Srinivasan, A. Gayasen, and T. Tuan, "Leakage control in FPGA routing fabric," in *Proc. Asia South Pacific Design Automation Conf.*, Shanghai, China, Jan. 2005, pp. 661–664.
[7] A. Gayasen, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and T. Tuan, "Reducing leakage energy in FPGAs using region-constrained placement," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, Monterey, CA, Feb. 2004, pp. 51–58.
[8] A. Lodi, L. Ciccarelli, and R. Giansante, "Combining low-leakage techniques for FPGA routing design," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, Monterey, CA, Feb. 2005, pp. 208–214.
[9] F. Li, Y. Lin, L. He, and J. Cong, "Low-power FPGA using pre-defined dual-vdd/dual-vt fabrics," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, Monterey, CA, Feb. 2004, pp. 42–50.
[10] F. Li, Y. Lin, and L. He, "FPGA power reduction using configurable dual-vdd," in *Proc. Design Automation Conf.*, San Diego, CA, Jun. 2004, pp. 735–740.
[11] Y. Lin, F. Li, and L. He, "Routing track duplication with fine-grained power-gating for FPGA interconnect power reduction," in *Proc. Asia South Pacific Design Automation Conf.*, Shanghai, China, Jan. 2005, pp. 645–650.
[12] A. Gayasen, K. Lee, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and T. Tuan, "A dual-VDD low power FPGA architecture," in *Proc. Int. Conf. Field-Programmable Logic Application*, Leuven, Belgium, Aug. 2004, pp. 145–157.
[13] F. Li, Y. Lin, and L. He, "Vdd programmability to reduce FPGA interconnect power," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 2004, pp. 760–765.
[14] J. H. Anderson and F. N. Najm, "Low-power programmable routing circuitry for FPGAs," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 2004, pp. 602–609.
[15] Y. Lin, F. Li, and L. He, "Power modeling and architecture evaluation for FPGA with novel circuits for vdd programmability," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, Monterey, CA, Feb. 2005, pp. 199–207.
[16] S. Yang, "Logic synthesis and optimization benchmarks, version 3.0," Microelectronics Center of North Carolina (MCNC), Research Triangle Park, Tech. Rep., 1991-IWLS-UG-Saeyang 1991.
[17] *International Technology Roadmap for Semiconductor*, (2003). [Online]. Available: http://public.itrs.net/
[18] R. Nair *et al.*, "Generation of performance constraints for layout," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 8, no. 8, pp. 860–874, Aug. 1989.
[19] G. Knol, D. Tellez, and M. Sarrafzadeh, "A delay budgeting algorithm ensuring maximum flexibility in placement," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 16, no. 11, pp. 1332–1341, Nov. 1997.
[20] C.-Y. Yeh and M. Marek-Sadowska, "Delay budgeting in sequential circuit with application on FPGA placement," in *Proc. Design Automation Conf.*, Anaheim, CA, Jun. 2003, pp. 202–207.
[21] ——, "Minimum-area sequential budgeting for FPGA," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, Nov. 2003, pp. 813–817.
[22] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Norwell, MA: Kluwer, Feb. 1999.
[23] Y. Lin, F. Li, and L. He, "Circuits and architectures for field programmable gate array with configurable supply voltage," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 13, no. 9, pp. 1035–1047, Sep. 2005.
[24] W. C. Elmore, "The transient response of damped linear networks with particular regard to wide-band amplifiers," *J. Appl. Phys.*, vol. 19, no. 1, pp. 55–63, Jan. 1948.
[25] University of Berkeley Device Group, *Predictive Technology Mode*, (2002). [Online]. Available: http://www.device.eecs.berkeley.edu/ptm/mosfet.html
[26] R. W. Brodersen, M. A. Horowitz, D. Markovic, B. Nikolic, and V. Stojanovic, "Methods for true power minimization," in *Proc. Int. Conf. Computer-Aided Design*, San Jose, CA, 2002, pp. 35–42.
[27] Y. Lin and L. He, "Leakage efficient chip-level dual-VDD assignment with time slack allocation for FPGA power reduction," UCLA Engr., Los Angeles, CA, Tech. Rep. 05-257, 2005.
[28] ILOG, Inc., *ILOG CPLEX Optimizers*. [Online]. Available: http://www.ilog.com/products/cplex/
[29] M. Berkelaar, *lp-solver: A Public Domain (MI)LP Solver*. [Online]. Available: http://groups.yahoo.com/group/lp_solve/
[30] M. Hamada *et al.*, "A top-down low power design technique using clustered voltage scaling with variable supply-voltage scheme," in *Proc. IEEE Custom Integrated Circuits Conf.*, Santa Clara, CA, 1998, pp. 495–498.
[31] D. Lewis *et al.*, "The Stratix routing and logic architecture," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, Monterey, CA, Feb. 2003, pp. 12–20.
[32] ——, "The Stratix II logic and routing architecture," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, Monterey, CA, Feb. 2005, pp. 14–20.
[33] *Xilinx Virtex-4 Device Family Datasheets*. [Online]. Available: http://www.xilinx.com/support/mysupport.htm



Yan Lin (S'05) received the B.E. degree in automation from Tsinghua University, Beijing, China, in 2002, the M.S. degree in electrical engineering from the University of California, Los Angeles (UCLA), in 2004, and is currently working toward the Ph.D. degree in the Electrical Engineering Department, UCLA.

His research interests include computer-aided design of very large scale integration circuits and systems, programmable fabrics, and low-power designs.

Lei He (S'94–M'99) received the Ph.D. degree in computer science from the University of California at Los Angeles (UCLA), in 1999.

He was a faculty member at the Electrical and Computer Engineering Department, University of Wisconsin, Madison, between 1999 and 2001. In 2002, he joined the Faculty of Electrical Engineering, UCLA. He has published over 100 technical papers. His research interests include modeling and design considering inductance effects and process variations for integrated circuits and packages, programmable logic fabrics and reconfigurable systems, and power-efficient circuits and systems.

Dr. He served as a TPC member of the Design Automation Conference, the International Symposium on Low Power Electronics and Design, and the International Symposium on Field Programmable Gate Array. He received the National Science Foundation CAREER Award, in 2000, UCLA Chancellor's Faculty Career Development Award (highest class), in 2003, an IBM Faculty Partner Award, in 2003, and a Northrop Grumman Excellence in Teaching Award, in 2005.