

# Dual-Vdd Interconnect with Chip-level Time Slack Allocation for FPGA Power Reduction

Yan Lin and Lei He

Electrical Engineering Department  
University of California, Los Angeles

## Abstract

To reduce FPGA power, Vdd programmability has been proposed recently to select Vdd-level for interconnects and to power-gate unused interconnects. However, Vdd-level converters used in the existing Vdd-programmable method consume a large amount of leakage. In this paper, we propose two ways to avoid using level converters in interconnects, *tree based level converter insertion (TLC)* and *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. We develop dual-Vdd assignment algorithms considering chip-level time slack allocation for maximum power reduction. Our algorithms include *TLC-S* and *dTLC-S*, 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 slightly higher power consumption but runs 3X faster. Compared to the existing *segment-based level converter insertion (SLC)* for dual-Vdd, dTLC-LP reduces interconnect power by 52.90% without performance loss for the MCNC benchmark circuits.

## Index Terms

Power\_minimization, Low-power\_design, Interconnect, Optimization.

## I. INTRODUCTION

FPGA power modeling and reduction has become an active research area recently. [1], [2] present power evaluation frameworks for generic parameterized FPGA architectures, and show that both interconnect and leakage power are significant for nanometer FPGAs. [3] presents a power-driven partition algorithm for mapping applications to FPGA with different Vdd-levels. [4] studies the interaction of a suite of power-aware FPGA CAD algorithms without changing the existing FPGAs. [5] proposes configuration inversion method to reduce leakage power of multiplexers without any additional hardware. In addition, low power FPGA circuits and architectures have been proposed. [6] designs a new type of routing multiplexer and proposes an input control method to reduce unused routing multiplexer leakage. [7] develops region-based power-gating for unused FPGA logic blocks to reduce leakage power. [8] studies low leakage interconnect circuitry, which combines gate biasing, body biasing and multi-threshold techniques to reduce interconnect leakage. [9], [10] are the first work introducing dual-Vdd and field programmability of Vdd to FPGA. [11] applies fine-grained power-gating to FPGA interconnects and presents a routing algorithm for pre-determined 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 to 38.73% of total power and 33.96% of total area respectively for the 20 largest MCNC benchmark circuits [16] at the ITRS 100nm technology node [17]. Power-gating unused level converters may reduce leakage but introduces extra area overhead<sup>1</sup>. Therefore it is less attractive compared to the methods to be presented in this paper to remove level converters.

Two existing approaches avoid directly using level converters in Vdd-programmable interconnects. [14] uses 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 an alternative level converter design, which has less leakage overhead but may cause larger delay and therefore larger short circuit power compared to the level converter

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

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%
diffcq	0.03603	7667824	128921	0.01856	51.52%	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	32.68%	2733309	35.01%
frisc	0.19169	46672107	944892	0.13606	70.98%	18038234	38.65%
misex3	0.08475	7548212	132440	0.01907	22.50%	2523180	33.43%
pdq	0.24585	43487207	878867	0.12656	51.48%	16781015	38.59%
s298	0.05599	9485182	157652	0.02270	40.55%	2991013	31.53%
s38417	0.26492	40827249	743341	0.10704	40.41%	14117128	34.58%
s38584.1	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%

TABLE I

LEAKAGE POWER AND AREA OF VDD-LEVEL CONVERTERS IN VDD-PROGRAMMABLE INTERCONNECTS WITH SEGMENT BASED LEVEL CONVERTER INSERTION (SLC) USING 100nm TECHNOLOGY. AREA IS IN NUMBER OF MINIMUM WIDTH TRANSISTORS. (LC STANDS FOR LEVEL CONVERTER.)

used in [13]. [12] only inserts a 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 Vdd-level for CLBs and interconnects. However, this path-based assignment lacks flexibility and cannot effectively leverage the surplus timing slack.

The primary contribution of this paper is to remove the 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. 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. 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  $T_{spec}$ , path timing constraints can be translated into the 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. [20], [21] apply delay budgeting to minimize Flip-Flops 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  $T_{spec}$ . 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 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 dTLC problem. TLC-S and dTLC-S implicitly allocate time slack first to interconnects with larger power sensitivity and assign VddL to them for more power reduction. dTLC-LP first explicitly allocate time slack to each routing tree by formulating the problem as an LP problem to maximize a lower bound of power reduction, and then 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 slightly higher power consumption but runs 3X faster. The best algorithm, dTLC-LP, reduces total interconnect power by 52.90% compared to the *segment based level converter insertion (SLC)* [13], and by 18.52% compared to the best approach proposed in [12].

The rest part of the paper is organized as follows. Section II presents the preliminaries and motivation. Section III introduces modeling and problem formulation. Section IV describes power sensitivity based algorithms and the LP based algorithm. Section V presents the experimental results and Section VI concludes this paper. The preliminary results of TLC was first presented in [15]<sup>2</sup>.

## II. BACKGROUND AND MOTIVATION

### A. Preliminaries

Interconnects consume most of the area and power of FPGAs. We assume the traditional island style routing architecture [22] as shown in Figure 1 in our study. The logic blocks (CLBs) are surrounded by routing channels consisting of wire segments.

<sup>2</sup>However, the emphasis of [15] is architecture evaluation considering Vdd programmability.

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 block* (see Figure 1 (b)). The multiplexer-based implementation chooses only one track in the routing channel and connects it to the CLB input pin. The buffers between the routing tracks and the multiplexer are *connection switches*. There is a *routing switch block* at each intersection of a horizontal channel and a vertical channel. The connections in a switch block (represented by the dashed lines in Figure 1 (c)) are programmable *routing switches*. We implement routing switches by tri-state buffers and use two tri-state buffers for each connection so that it can be programmed independently for either direction. An *interconnect switch* is either a routing switch or a connection switch. An *interconnect segment* is a wire segment driven by an interconnect switch. As suggested in [22], we assume a uniform length 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.

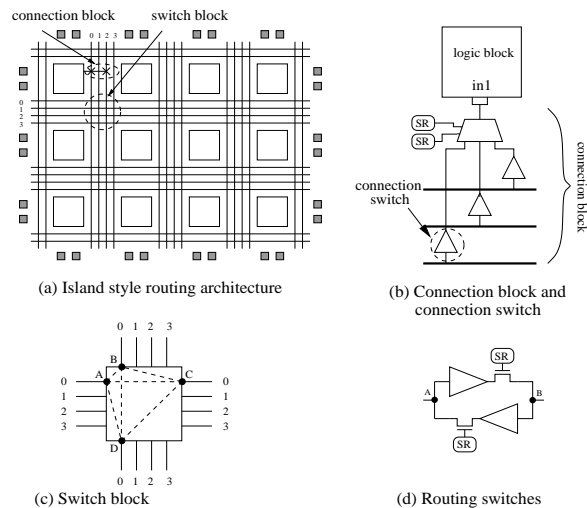


Fig. 1. (a) Island style routing architecture; (b) Connection block; (c) Switch block; (d) Routing switches. (SR stands for SRAM cell.)

Vdd programmability can be applied to interconnects to reduce FPGA power. Figure 2 shows the Vdd-programmable interconnect switch proposed in [13]. For the Vdd-programmable routing switch in Figure 2 (a), two PMOS power transistors M3 and M4 are inserted between the tri-state buffer and VddH, VddL power rails, respectively. Turning off one of the power transistors 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 [13]. There are power and delay overhead associated with the power transistors insertion. The dynamic power overhead is almost negligible. This is because that the 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. Another type of routing resources is the connection block in Figure 2 (b). Similar to the routing switch, programmable-Vdd is also applied to the connection switch.

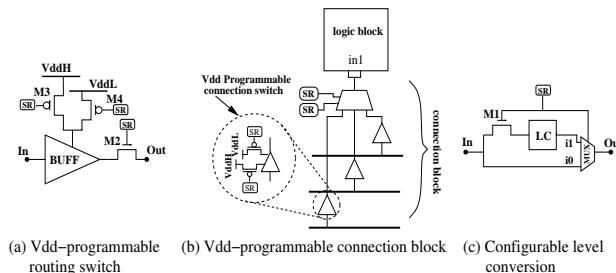


Fig. 2. (a) Vdd-programmable routing switch; (b) Vdd-programmable connection block; (c) Configurable Vdd-level conversion. (SR stands for SRAM cell and LC stands for level converter.)

A Vdd-level converter is needed whenever a VddL interconnect switch drives a VddH interconnect switch to avoid excessive leakage. In other cases, the level converter can be bypassed. As shown in Figure 2 (c), a pass transistor M1 and a MUX together with a configuration SRAM cell can be used to implement a configurable level conversion. A configurable level conversion circuit is inserted in front of each interconnect switch to provide fine-grained Vdd programmability for interconnects in [13]. 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. For the rest part of the paper, we use switch to represent interconnect switch for simplicity whenever there is no ambiguity.

### B. Motivation

The Vdd-programmable interconnects proposed in [13] insert a configurable Vdd-level converter in front of each interconnect switch. This can be called 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. Analysis (see Table I) shows that the average leakage and area due to the level converters in routing channels represent 38.73% of total power and 33.96% of total area for MCNC benchmark circuits. *If CAD algorithms can guarantee that no VddL interconnect switch drives VddH switches, no level converter is needed.* In this paper, we propose two ways to avoid using level converters in interconnects. In the first approach, we enforce that there is only one Vdd-level within each routing tree, namely, *tree based level converter insertion (TLC)*. Since two routing trees will not intersect with each other, we do not need level converters in routing channels. Figure 3 (a) shows the TLC and illustrates the situation that a VddH routing tree and VddL routing tree can share a same routing track without level converters in routing channels. In the second approach, different Vdd-levels can exist within a routing tree, but no VddL switch drives VddH switches, namely, *dual-Vdd tree based level converter insertion (dTLC)*. As shown in Figure 3 (b), we allow a VddH switch to drive VddL switches within one routing tree for dTLC. 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. 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.

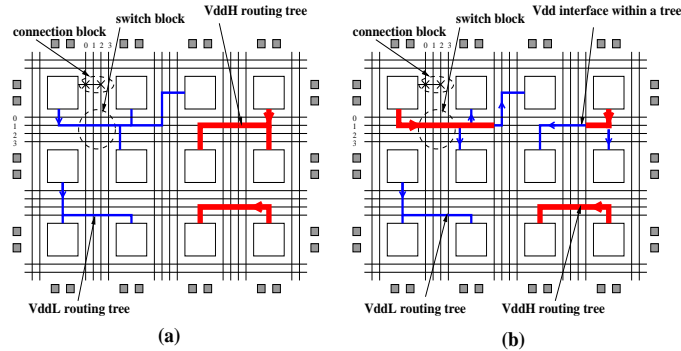


Fig. 3. (a) Tree based level converter insertion (TLC); (b) Dual-Vdd tree based level converter insertion (dTLC).

$\mathcal{G}(\mathcal{V}, \mathcal{E})$	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 $\mathcal{R}_i$

TABLE II

NOTATIONS FREQUENTLY USED IN THIS PAPER.

## III. MODELING AND PROBLEM FORMULATION

### A. 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 which have no incoming edges, and  $\mathcal{PO}$  to represent the set of primary outputs and register inputs which have no outgoing edges.

Elmore delay model [23] is used to calculate the routing delay. We define the *fanout cone* of an interconnect switch as the sub-tree of the routing tree rooted at the switch. Assigning VddL to a switch affects the delay from 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 pre-characterize the intrinsic delay and effective driving resistance for a switch under VddH and VddL, respectively. We use Berkeley predictive device model [24] at the ITRS 100nm 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.

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

TABLE III

THE CALIBRATED INPUT AND LOAD CAPACITANCE OF A 7X BUFFER USING SPICE UNDER 1.3V AND 0.8V, RESPECTIVELY.

### B. Power Modeling with Dual-Vdd

There are three power sources in FPGAs, switching power, short-circuit power and 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 change 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 of simplification in the modeling, a more accurate power simulation will be performed to verify experimental results in Section V.

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

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

where  $N_r$  is the total number of routing trees,  $f_s(i)$  is the transition density of  $i^{th}$  routing tree  $\mathcal{R}_i$ ,  $N_s(i)$  is the number of switches in  $\mathcal{R}_i$ , and  $Vdd_{ij}$ ,  $P_s(Vdd_{ij})$  and  $c$  are the Vdd-level, leakage power and load capacitance of each switch respectively. For the rest part of the paper, we use  $\mathcal{R}_i$  to represent  $i^{th}$  routing tree. The dynamic power quadratically depends on the Vdd-level while the 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 Vdd-level of  $j^{th}$  switch in  $\mathcal{R}_i$  as follows

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

Reducing the Vdd-level can reduce both dynamic and leakage power. The interconnect power reduction  $P_r$  using programmable dual-Vdd can be expressed as the sum of dynamic power reduction and leakage power reduction as follows,

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

where  $N_l(i)$  is the number of VddL switches that can be achieved in  $\mathcal{R}_i$  and  $\Delta P_s$  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.

### C. Problem Formulation

Removing Vdd-level converters requires that no VddL switch should drive VddH switches. For TLC, only one Vdd-level can be used within each routing tree, and the Vdd-level constraints can be expressed as

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

i.e., each pair of switches within a routing tree have the same Vdd-level. For dTLC, we can have different Vdd-levels within one routing tree, and the Vdd-level constraints can be expressed as

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

i.e., no VddL switch should drive VddH switches.  $\mathcal{FO}_{ij}$  gives the set of fanout switches of  $j^{\text{th}}$  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 which partitions the constraints on path delay into constraints on delay across circuit elements or routing. Let  $a(v)$  be the arrival time for vertex  $v$  in  $\mathcal{G}$  and the timing constraints become

$$a(v) \leq T_{spec} \quad \forall v \in \mathcal{PO} \quad (5)$$

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

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

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 below objective function (8) is to maximize the power reduction (2).

$$\text{Maximize} \sum_{i=0}^{N_r-1} [0.5 f_{clk} \cdot c \cdot \Delta Vdd^2 \cdot f_s(i) + \Delta P_s] \cdot N_i(i) \quad (8)$$

The TLC problem consists of objective function(8), Vdd-level constraints (3) and timing constraints (5), (6) and (7). The dTLC problem is same as the tree based problem except that Vdd-level constraints (4) replace (3).

## IV. CHIP-LEVEL VDD ASSIGNMENT

### A. Tree Based Level Converter Insertion (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  $\Delta P / \Delta Vdd$ , which is defined as the power reduction ( $\Delta P$ ) divided by the Vdd-level difference ( $\Delta Vdd$ ) between VddH and VddL, for each switch with the wire it drives. The 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  $\sum_{j=0}^{N_s(i)-1} \Delta P_{ij} / \Delta Vdd$ , where  $\Delta P_{ij} / \Delta Vdd$  is the power sensitivity of  $j^{\text{th}}$  switch in  $\mathcal{R}_i$ .

A greedy algorithm similar to that in [25] is performed to assign Vdd-level for routing trees (See Figure 4). 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 re-visited in subsequent iterations. After the dual-Vdd assignment, we obtain a dual-Vdd netlist without performance loss.

**Tree based algorithm:**  
Assign VddH to all routing trees and mark them as ‘untried’;  
Calculate power-sensitivity for all routing trees;  
While(  $\exists$  ‘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. 4. Sensitivity based algorithm TLC-S for TLC problem.

### B. Dual-Vdd Tree Based Level Converter Insertion (dTLC)

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

1) *Sensitivity Based Algorithm (dTLC-S)*: The sensitivity based algorithm dTLC-S is quite similar to TLC-S except 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 into two categories. In the first category, a candidate switch is ‘untried’, and it 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 the 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 that is summarized in Figure 5.

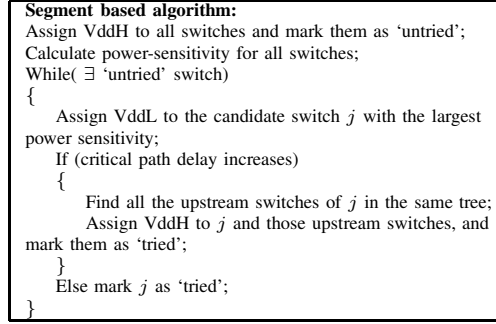


Fig. 5. Sensitivity based algorithm dTLC-S for dTLC problem.

2) *Linear Programming 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. Below, we present a linear programming 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 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 refinement to leverage surplus time slack. The details are discussed below.

#### • Chip-level Time Slack Allocation

##### Estimation for Number of Low-Vdd Switches

The *slack*  $s_{ij}$  of a connection between the source and  $j^{th}$  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  $\Delta d$ , where  $\Delta d$  is the delay increase for an interconnect segment by changing the Vdd-level from VddH to VddL. Figure 6 presents a 2-sink routing tree as an example.  $S_0$  and  $S_1$  are the slacks allocated to *Sink0* and *Sink1*, respectively. In Figure 6 (a), VddL can be assigned to  $b_2$  given  $S_0 = 1$  and VddL can be assigned to  $b_3$  given  $S_1 = 1$ . When we increase the slack  $S_1$  for *Sink1* to 2 in Figure 6 (b),  $b_0$  has to stay VddH restricted by  $S_0 = 1$ . In other words,  $b_0$  is restricted by both  $S_0$  and  $S_1$ , and VddL can only be assigned to  $b_0$  when  $S_0 \geq 3 \wedge S_1 \geq 2$ . Figure 6 (c) shows the case in which VddL is assigned to all the switches given  $S_0 = 3 \wedge S_1 = 2$ . 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, 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 part of the paper, we use slack to represent the useful slack.

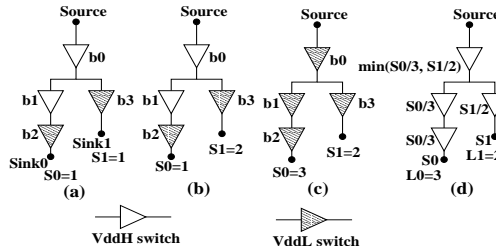


Fig. 6. An example for estimating number of VddL switches.

Figure 6 (a) and (b) show that we may achieve the same number of VddL switches with different slacks. Given a routing tree with 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  $k^{th}$  sink in  $\mathcal{R}_i$ . We define *sink list*

$\mathcal{SL}_{ij}$  as the set of sinks in the fanout cone of  $j^{th}$  switch in  $\mathcal{R}_i$ . We then estimate the number of VddL switches given the allocated slack as

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

To estimate the number of VddL switches in tree  $\mathcal{R}_i$ , we first deliberately distribute the slack  $s_{ik}$  evenly to the  $l_{ik}$  switches in the path from source to  $k^{th}$  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  $k^{th}$  sink with minimum  $s_{ik}/l_{ik}$  in sink list  $\mathcal{SL}_{ij}$  as the most critical sink to  $j^{th}$  switch in  $\mathcal{R}_i$ . Figure 6 (d) gives an example and the estimated number of VddL switches is calculated as

$$F_n = S0/3 + S0/3 + S1/2 + \min(S0/3, S1/2)$$

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

It is easy to see that (9) gives the exact number of VddL switches for 1-sink tree. For a 2-sink tree, we can verify that (9) gives a lower bound of 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 is presented in the appendix.

### LP Problem Formulation

The objective function (8) is to maximize power reduction which is the weighted sum of 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  $j^{th}$  switch in  $\mathcal{R}_i$  and some additional constraints. The new objective function after transformation plus the *additional constraints* can be expressed as

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

s.t.

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

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

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

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

The slack upper bound constraints can be expressed as

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

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

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

where vertex  $p_{i0}$  is the source of  $\mathcal{R}_i$  in  $\mathcal{G}$ , vertex  $p_{ik}$  is  $k^{th}$  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 the edges other than routing in  $\mathcal{G}$ , the constraints can be expressed as

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

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 a linear programming (LP) problem.



There are well-developed linear programming solvers available from both the commercial world like [26] and the academia like [27]. In this paper, we use the LP solver from [27]. For the rest part of the paper, we use LP problem to represent the time slack allocation problem.

- **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 Figure 7). 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  $\mathcal{R}_i$ .

```

Bottom-up assignment within  $\mathcal{R}_i$ :
Assign VddH to all switches in  $\mathcal{R}_i$  and mark them as ‘untried’;
While(  $\exists$  candidate switch)
{
  Assign VddL to a candidate switch  $j$ ;
  If (timing constraints violated)
  {
    Find all the upstream switches of  $j$  in  $\mathcal{R}_i$ ;
    Assign VddH to  $j$  and those upstream switches, and
    mark them as ‘tried’;
  }
  Else mark  $j$  as ‘tried’;
}

```

Fig. 7. Net-level bottom-up assignment.

*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  $j^{th}$  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  $j^{th}$  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.  $\square$

*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 <sup>3</sup>, and 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 Vdd-level of the VddL switch and its fanout VddH switches without violating timing constraints and achieve 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 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.  $\square$

- **Refinement**

After net-level assignment, we may further reduce power by leveraging surplus slack. Figure 6(b) shows a routing tree containing surplus slack.  $b_0$  has to stay VddH due to the fact that it is restricted by  $S_0 = 1$ . Therefore, *Sink1* can only consume one unit slack from  $S_1$  and there is 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 Figure 5) to achieve more VddL switches and further reduce power.

## V. EXPERIMENTAL RESULTS

In this section, we conduct experiments on the MCNC benchmark set. We first compare the interconnect power and runtime between the sensitivity based algorithms *TLC-S*, *dTLC-S* and the 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, *h2ILCi* [12], to the baseline using Vdd-programmable interconnects with segment based Vdd-level converter insertion (*SLC*) [13]. We use the same Vdd-programmable logic blocks and interconnects in [13], but no level converter is 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 ~ 0.7 suggested in [28], we use 1.3v for VddH and 0.8v for VddL in our experiments at 100nm technology node.

We first use VPR [22] for single-Vdd placement and routing. Before applying *SLC*, *TLC-S*, *dTLC-S* or *dTLC-LP* to the Vdd-programmable interconnects, a sensitivity based assignment [10] is first performed to assign Vdd-level for Vdd-programmable

<sup>3</sup>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.

logic blocks without performance loss<sup>4</sup>. It is 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 the 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. Column 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 that TLC uses a routing tree as the assignment unit and does not allow the mix of different Vdd-levels within a routing tree.

Column 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 column 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 that dTLC-LP considers both the global and local optimality. Figure 8 compares the estimated VddL switch number given by (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.

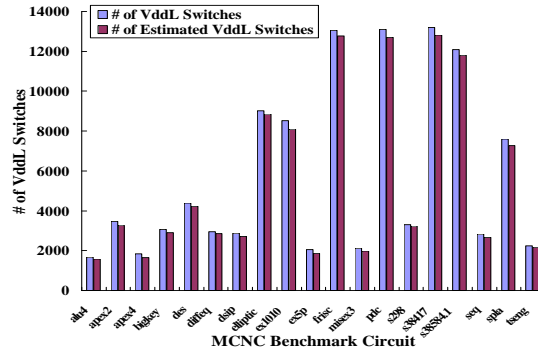


Fig. 8. Comparison between VddL switch number achieved by dTLC-LP before refinement and the estimated VddL switch number.

For dTLC-LP, we also present the contribution of refinement step in column 4, column 7, column 10 and column 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, respectively. It is clear that the refinement step is effective to re-distribute 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  $\Delta d$ ). Second, the objective of the 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<sup>5</sup> between the three algorithms. TLC-S is the fastest among the three algorithms. dTLC-S and dTLC-LP take 1.64X and 5.10X 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 the MCNC benchmark circuits have already been partitioned into combinational circuit blocks. In general, large circuits might be partitioned and the LP problem then might be solved for each partition to reduce runtime. Compared to dTLC-LP, dTLC-S has slightly larger power consumption, but runs 3X faster and is effective for large circuits. dTLC-LP is worthwhile for small circuits and can achieve best power reduction.

<sup>4</sup>All the algorithms SLC, TLC-S, dTLC-S and dTLC-LP do consider the fact that VddL logic blocks consume time slack.

<sup>5</sup>The runtime includes single-Vdd placement and routing by VPR and generating the interface files between VPR and *fpgaEva-LP2*.

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%)
diffsq	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%)

TABLE IV

PERCENTAGE OF VDDL SWITCHES AND INTERCONNECT POWER ACHIEVED BY 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
diffsq	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		1X	1.64X	5.10X

TABLE V

RUNTIME COMPARISON BETWEEN TLC-S, dTLC-S AND dTLC-LP.

### 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 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 algorithm *dTLC-S*, *dTLC-S-I* and *dTLC-S-U*, all implemented based on power sensitivity. Column 2-7 in the Table 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 percentage of VddL CLBs and switches. Column 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, h2ILCi 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 *h2ILCi* [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. [12] inserts a level converter either at each CLB input or output. A path-based assignment is performed for the Vdd-programmable interconnects. It has been shown that *h2ILCi*, which inserts a level converter at each CLB input and initializes all the circuit elements with VddH, achieves the lowest power consumption among all proposed approaches in [12].

Circuit	% of VddL CLBs			% of VddL switches			total power		
	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%

TABLE VI

PERCENTAGE OF VDDL CLBs AND INTERCONNECT SWITCHES, AND TOTAL POWER ACHIEVED BY THREE APPROACHES OF dTLC-S.

1) *Interconnect Power Comparison with Relaxed Timing Specification*: The 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. Figure 9 compares the percentage of VddL switches and relaxed critical path delay tradeoff curves achieved by SLC, h2ILCi and dTLC-LP. The VddL switch percentage and critical path delay are the arithmetic and geometric mean over the 20 largest MCNC benchmark circuits, respectively. When the timing specification is not relaxed, SLC, h2ILCi and dTLC-LP achieve 74.79%, 41.98% and 77.70%, respectively. Both of SLC and dTLC-LP achieve more VddL switches than h2ILCi. It is because that all the trees driven by one CLB have the same Vdd-level as the source CLB and VddH circuit element is not allowed to drive VddL circuit element without the presence of level converter in h2ILCi. dTLC-LP consistently achieves the highest percentage of VddL switches compared to previous approach SLC<sup>6</sup> and h2ILCi at different relaxed delays.

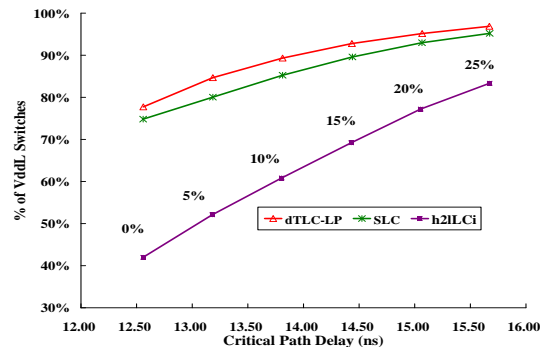


Fig. 9. Percentage of VddL interconnect switches vs. critical path delay curves for SLC, h2ILCi and dTLC-LP.

Figure 10 compares the interconnect power and critical path delay tradeoff curves achieved by SLC, h2ILCi and dTLC-LP. Both the interconnect power and critical path delay are the geometric mean over the 20 largest MCNC benchmark circuits. Compared to SLC, h2ILCi and dTLC-LP reduce interconnect power by 41.44% and 52.90% without relaxing timing specification, respectively. Compared to h2ILCi, dTLC-LP reduces interconnect power by 18.52% at the highest clock frequency. dTLC-LP consistently achieves the lowest power compared to SLC and h2ILCi at different relaxed delays. Compared to SLC at the same relaxed delay, the LP based algorithm achieves 89.31% VddL switches and reduces interconnect power by 58.18% when we relax critical path delay by 10%. The power gap between dTLC-LP and h2ILCi decreases at larger relaxed delay.

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

This is because that VddL eventually can be assigned to all switches (see Figure 9) and interconnect power reduction will saturate if we allow sufficient critical path increase.

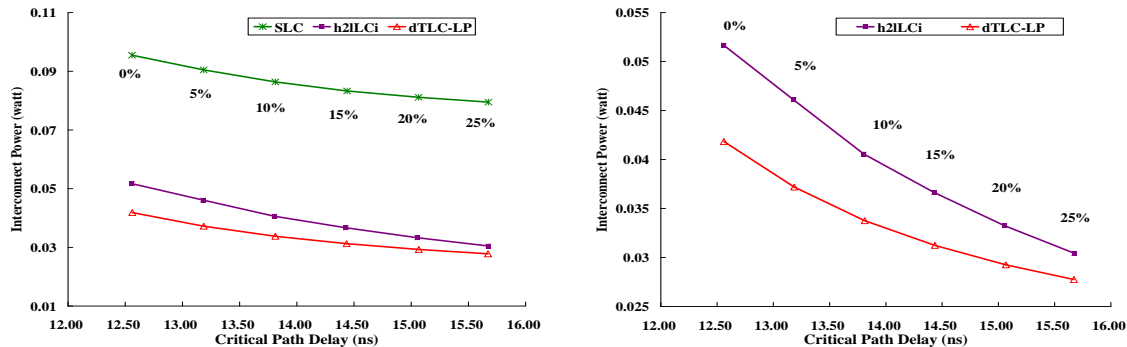


Fig. 10. Interconnect power and critical path delay tradeoff for SLC, h2ILCi and dTLC-LP.

2) *Area Comparison*: Table VII present the FPGA total area given by single-Vdd, SLC, h2ILCi and dTLC-LP, respectively. We use the same area model from [22], in which area is counted in number of minimum width transistor areas with 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:

$$Area(W) = 0.5 + \frac{W}{2 \cdot W_{min}} \quad (17)$$

As presented in the table, the area overhead given by SLC, h2ILCi and dTLC-LP are 151.33%, 65.02% and 65.53% compared to the single-Vdd FPGA, respectively. h2ILCi 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.

circuit	single-Vdd area	SLC		h2ILCi		dTLC-LP	
		area	overhead	area	overhead	area	overhead
alu4	2925831	7136109	143.90%	4779206	63.35%	4795092	63.89%
apex2	4384857	11078412	152.65%	7215910	64.56%	7237060	65.05%
apex4	3007921	7534174	150.48%	4903820	63.03%	4917356	63.48%
bigkey	8614605	19628149	127.85%	14192729	64.75%	14261255	65.55%
clma	36361942	100634278	176.76%	61402657	68.87%	61531343	69.22%
des	12371141	28245400	128.32%	20338979	64.41%	20435235	65.18%
diffreq	3185418	7667824	140.72%	5199431	63.23%	5217855	63.80%
dsip	9718346	22792498	134.53%	15986128	64.49%	16054654	65.20%
elliptic	9056268	23394478	158.32%	15004948	65.69%	15046402	66.14%
ex1010	11605734	30506887	162.86%	19308966	66.37%	19358692	66.80%
ex5p	3106231	7808090	151.37%	5061245	62.94%	5074781	63.37%
frisc	17028484	46672107	174.08%	28575123	67.81%	28633873	68.15%
misex3	3063483	7548212	146.39%	5009146	63.51%	5025032	64.03%
pdic	15929127	43487207	173.00%	26652048	67.32%	26706192	67.66%
s298	3947796	9485182	140.27%	6470105	63.89%	6494169	64.50%
s38417	15975183	40827249	155.57%	26625521	66.67%	26710121	67.20%
s38584	13005036	33144130	154.86%	21637358	66.38%	21705884	66.90%
seq	4384857	11078412	152.65%	7215910	64.56%	7237060	65.05%
spla	9664508	25585907	164.74%	16038715	65.95%	16076315	66.34%
tseng	2275254	5397201	137.21%	3698578	62.56%	3712114	63.15%
avg.	-	-	151.33%	-	65.02%	-	65.53%

TABLE VII

FPGA AREA COMPARISON BETWEEN SINGLE-VDD, SLC, h2ILCi AND dTLC-LP. AREA IS IN NUMBER OF MINIMUM WIDTH TRANSISTORS. WE USE 210X AND 4X PMOS POWER TRANSISTORS FOR CLBS AND 7X SWITCHES, RESPECTIVELY.

## VI. CONCLUSIONS

To remove Vdd-level converters in routing channels that introduce large leakage and area overhead, we have proposed *tree based level converter insertion (TLC)*, where there is only one Vdd-level within each routing tree. We have also proposed *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. 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.

We have developed a few Vdd-level assignment algorithms considering time slack allocation to maximize power reduction. Our Vdd-level assignment algorithms include power sensitivity based algorithms, *TLC-S* and *dTLC-S* for TLC and dTLC problem respectively, and a linear programming (LP) based algorithm *dTLC-LP* for dTLC problem. *TLC-S* and *dTLC-S* implicitly allocate time slack first to interconnects with higher power sensitivity and assign VddL to them for more power reduction. *dTLC-LP* first explicitly allocates time slack to each routing tree using linear programming, and then assigns 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 slightly higher power consumption but runs 3X 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 total interconnect power by 52.90% compared to the *segment based level converter insertion (SLC)* proposed in [13], and by 18.52% compared to the best approach *h2ILCi* proposed in [12] where all routing trees driven by one CLB have the same Vdd-level as the source CLB.

The state-of-the-art commercial FPGAs have used wire segments of different lengths and applied uni-directional level-restore routing switch in routing architecture [29]–[31] to improve performance. Our methodology proposed in this paper can be extended to the interconnects with these novel features. We believe that dTLC formulation can still outperform TLC and h2ILCi 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 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 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 this is slightly better than (or as good as) allocating time slack in a uniform fashion to both interconnects and CLBs. In the future, such comparison may be re-conducted using linear programming based algorithm for the three ways of allocating time slack. In our study, we perform Vdd-level assignment based on single-Vdd routing, which may be sub-optimal for Vdd-programmable interconnects. In the future, we will also study power-driven routing, which simultaneously performs routing and Vdd-level assignment for Vdd-programmable interconnects.

#### ACKNOWLEDGEMENT

This paper is partially supported by NSF CAREER award CCR-0093273, and NSF grant CCR-0306682. Comments can be addressed to [lhe@ee.ucla.edu](mailto:lhe@ee.ucla.edu). The authors like to thank Dr. Fei Li and Mr. Jinjun Xiong at UCLA for helpful discussions.

#### REFERENCES

- [1] K. Poon, A. Yan, and S. Wilton, "A flexible power model for FPGAs," in *Proc. of 12th International conference on Field-Programmable Logic and Applications*, Sep 2002.
- [2] F. Li, D. Chen, L. He, and J. Cong, "Architecture evaluation for power-efficient FPGAs," in *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, Feb 2003.
- [3] R. Mukherjee and S. O. Memik, "Power-driven design partitioning," in *Proc. Intl. Conf. Field-Programmable Logic and its Application*, August 2004.
- [4] J. Lamoureux and S. J. Wilton, "On the interaction between power-aware FPGA CAD algorithms," in *Proc. Intl. Conf. Computer-Aided Design*, November 2003, pp. 701–708.
- [5] J. H. Anderson, F. N. Najm, and T. Tuan, "Active leakage power optimization for FPGAs," in *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2004.
- [6] S. Srinivasan, A. Gayasen, and T. Tuan, "Leakage control in FPGA routing fabric," in *Proc. Asia South Pacific Design Automation Conf.*, January 2005.
- [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 Intl. Symp. Field-Programmable Gate Arrays*, February 2004.
- [8] A. Lodi, L. Ciccarelli, and R. Giansante, "Combining low-leakage techniques for FPGA routing design," in *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2005.
- [9] F. Li, Y. Lin, L. He, and J. Cong, "Low-power FPGA using pre-defined dual-vdd/dual-vt fabrics," in *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2004.
- [10] F. Li, Y. Lin, and L. He, "FPGA power reduction using configurable dual-vdd," in *Proc. Design Automation Conf.*, June 2004.
- [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.*, Jan 2005.
- [12] A. Gayasen, K. Lee, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and T. Tuan, "A dual-vdd low power FPGA architecture," in *Proc. Intl. Conf. Field-Programmable Logic and its Application*, August 2004.
- [13] Fei Li and Yan Lin and Lei He, "Vdd programmability to reduce FPGA interconnect power," in *Proc. Intl. Conf. Computer-Aided Design*, November 2004.
- [14] Jason H. Anderson and Farid N. Najm, "Low-power programmable routing circuitry for FPGAs," in *Proc. Intl. Conf. Computer-Aided Design*, November 2004.
- [15] Y. Lin, F. Li, and L. He, "Power modeling and architecture evaluation for FPGA with novel circuits for vdd programmability," in *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, February 2005.
- [16] S. Yang, "Logic synthesis and optimization benchmarks, version 3.0," Microelectronics Center of North Carolina (MCNC), Tech. Rep., 1991.
- [17] International Technology Roadmap for Semiconductor, in <http://public.itrs.net/>, 2003.
- [18] R. Nair and et al, "Generation of performance constraints for layout," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, pp. 860–874, Aug. 1989.
- [19] G. Knol, D. Tellez and M. Sarrafzadeh, "A delay budgeting algorithm ensuring maximum flexibility in placement," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, pp. 1332–1341, 1997.

- [20] C.-Y. Yeh and M. Marek-Sadowska, "Delay budgeting in sequential circuit with application on FPGA placement," in *Proc. Design Automation Conf.*, June 2003.
- [21] —, "Minimum-area sequential budgeting for FPGA," in *Proc. Intl. Conf. Computer-Aided Design*, Nov. 2003.
- [22] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, Feb 1999.
- [23] W. C. Elmore, "The transient response of damped linear networks with particular regard to wide-band amplifiers," *Journal of Applied Physics*, vol. 19, no. 1, pp. 55–63, Jan. 1948.
- [24] U. of Berkeley Device Group, "Predictive technology model," in <http://www.device.eecs.berkeley.edu/ptm/mosfet.html>, 2002.
- [25] R. W. Brodersen, M. A. Horowitz, D. Markovic, B. Nikolic, and V. Stojanovic, "Methods for ture power minimization," in *Proc. Intl. Conf. Computer-Aided Design*, 2002, pp. 35–42.
- [26] *ILOG CPLEX optimizers*. <http://www.ilog.com/products/cplex/>.
- [27] M Berkelaar, *lp-solver: a public domain (MI)LP solver*. [ftp://ftp.ics.ele.tue.nl/pub/lp\\_solve/](ftp://ftp.ics.ele.tue.nl/pub/lp_solve/).
- [28] M. Hamada and 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.*, 1998, pp. 495–498.
- [29] D. Lewis and et al, "The stratix routing and logic architecture," in *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, Feb 2003.
- [30] —, "The stratix ii routing and logic architecture," in *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, Feb 2005.
- [31] "Xilinx product datasheets," in <http://www.xilinx.com/literature>.
- [32] Y. Lin and L. He, "Leakage efficient chip-level dual-vdd assignment with time slack allocation for FPGA power reduction," in *Proc. Design Automation Conf.*, June 2005.

## APPENDIX

### proof of Theorem 1:

For a one sink routing tree  $\mathcal{R}$  containing  $N_s$  switches as shown in Figure 11, given the allocated slack  $s_1$  for *Sink1*, the number of VddL switches that can be achieved is  $N = s_1$ . According to (9), the estimated VddL switch number is  $F = \sum_{j=0}^{N_s-1} \frac{s_1}{l_1} = s_1$ , where  $l_1 = N_s$  as there is only one sink in  $\mathcal{R}$ . It is obvious that  $F \leq N$ , and therefore (9) gives a lower bound of VddL switch number that can be achieved for 1-sink tree  $\mathcal{R}$ .

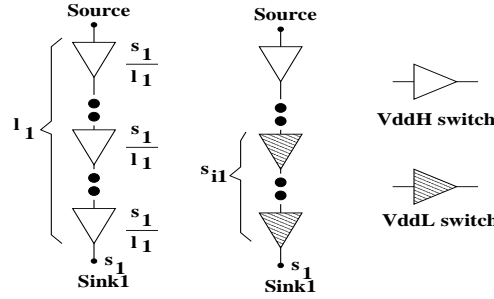


Fig. 11. A 1-sink routing tree  $\mathcal{R}$ .

We will prove Theorem 1 by induction. Suppose Theorem 1 holds for any tree  $\mathcal{R}_{n-1}$  with  $n-1$  sinks, we are to prove that is true for any  $n$ -sink tree  $\mathcal{R}_n$ . Let  $s_k$  denote the allocated slack for  $k^{th}$  sink in  $\mathcal{R}_n$ , and  $l_k$  denote the number of switches in the path from the source to  $k^{th}$  sink in  $\mathcal{R}_n$ . Without loss of generality, we arrange the sink order such that for  $n^{th}$  sink,

$$\frac{s_n}{l_n} = \min\left(\frac{s_k}{l_k} : \forall 1 \leq k \leq n\right) \quad (18)$$

We trace back  $\mathcal{R}_n$  from  $n^{th}$  sink and find the first branching point. Let  $\hat{b}_n$  be the branch from the immediate downstream switch of the branching point to  $n^{th}$  sink, and  $\hat{l}_n$  denote the number of switches in  $\hat{b}_n$ . Note that the switches in  $\hat{b}_n$  are restricted only by  $s_n$ . Suppose we remove the branch  $\hat{b}_n$  from  $\mathcal{R}_n$  and keep the allocated slacks for the remaining  $n-1$  sinks unchanged, we can get a sub-tree  $\mathcal{R}_{n-1}$  with  $n-1$  sinks. VddL may or may not be assigned to the immediate upstream switch of the removed branch  $\hat{b}_n$  in  $\mathcal{R}_{n-1}$ . We discuss these two situations as following.

Figure 12 shows the situation in which VddL cannot be assigned to the switch that drives the branch  $\hat{b}_n$  in  $\mathcal{R}_{n-1}$ . As there is no level converter in  $\mathcal{R}_n$ , all the upstream switches of  $\hat{b}_n$  have to stay VddH. Note that VddL cannot be assigned to the upstream switches of  $\hat{b}_n$  regardless of  $s_n$  as they are restricted by the slacks other than  $s_n$ . Let  $N_n$  and  $F_n$  denote number and estimated number of VddL switches that can be achieved in  $\mathcal{R}_n$ , respectively. Let  $N_{n-1}$  and  $F_{n-1}$  denote number and estimated number of VddL switches that can be achieved in the subtree  $\mathcal{R}_{n-1}$ , respectively. If  $s_n < \hat{l}_n$  (see Figure 12(a)), we can assign VddL to the bottom  $s_n$  switches in  $\hat{b}_n$ , and we have

$$\begin{aligned} F_n &= F_n|_{\mathcal{R}_{n-1}} + F_n|_{\hat{b}_n} = F_n|_{\mathcal{R}_{n-1}} + \frac{s_n}{l_n} \cdot \hat{l}_n \\ &\leq F_{n-1} + s_n \leq N_{n-1} + N_n|_{\hat{b}_n} \\ &= N_n|_{\mathcal{R}_{n-1}} + N_n|_{\hat{b}_n} = N_n \end{aligned} \quad (19)$$

where  $F_n|_{\mathcal{R}_{n-1}}$  and  $F_n|_{\hat{b}_n}$  are the estimated VddL switch number in  $\mathcal{R}_{n-1}$  and  $\hat{b}_n$  considering all the slacks including  $s_n$ , respectively.  $F_n|_{\mathcal{R}_{n-1}} \leq F_{n-1}$  because of (18). Therefore, considering  $s_n$  could only decrease or maintain the slacks distributed

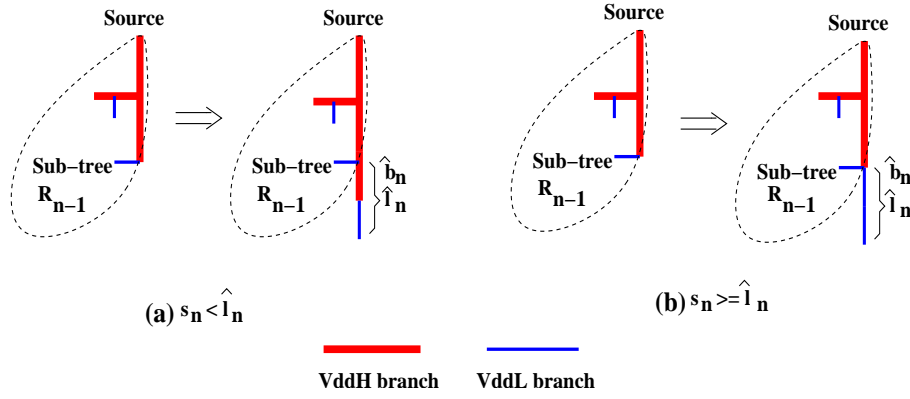


Fig. 12. A  $n$ -sink routing tree  $\mathcal{R}_n$  in which VddL cannot be assigned to the switch that drives  $\hat{b}_n$  in  $\mathcal{R}_{n-1}$ .

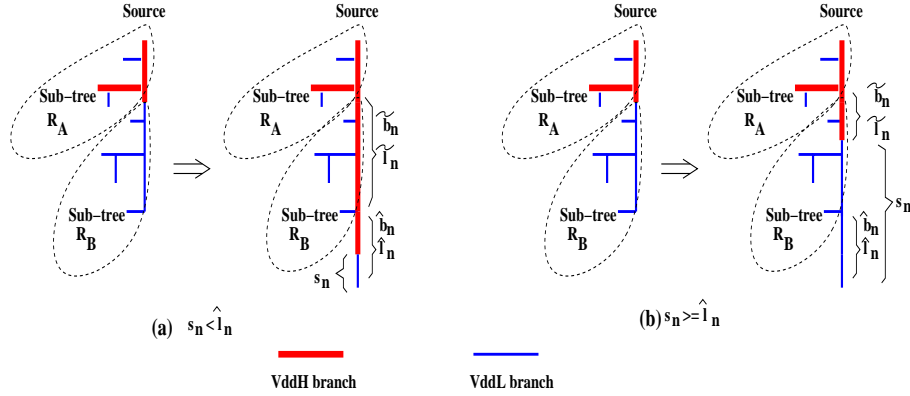


Fig. 13. A  $n$ -sink routing tree  $\mathcal{R}_n$  in which VddL can be assigned to the switch that drives  $\hat{b}_n$  in  $\mathcal{R}_{n-1}$ .

to the switches in  $\mathcal{R}_{n-1}$ . We have known  $F_{n-1} \leq N_{n-1}$  by induction.  $N_{n-1} = N_n|_{\mathcal{R}_{n-1}}$  because  $s_n \leq \hat{l}_n$  and therefore  $s_n$  does not affect the Vdd-level assignment in  $\mathcal{R}_{n-1}$ . It is clear that  $\frac{s_n}{l_n} \cdot \hat{l}_n \leq s_n$  as  $\hat{l}_n \leq l_n$ .

Similarly, if  $s_n \geq \hat{l}_n$  (see Figure 12(b)), VddL can be assigned to all the switches in  $\hat{b}_n$  while the upstream switches of  $\hat{b}_n$  have to stay VddH, and we have

$$\begin{aligned}
 F_n &= F_n|_{\mathcal{R}_{n-1}} + F_n|_{b_n} = F_n|_{\mathcal{R}_{n-1}} + \frac{s_n}{l_n} \cdot \hat{l}_n \\
 &\leq F_{n-1} + \hat{l}_n \leq N_{n-1} + N_n|_{\hat{b}_n} \\
 &= N_n|_{\mathcal{R}_{n-1}} + N_n|_{\hat{b}_n} = N_n
 \end{aligned} \tag{20}$$

$\frac{s_n}{l_n} \cdot \hat{l}_n \leq \hat{l}_n$  because of the upper bound constraint of useful slack, i.e.,  $s_n \leq l_n$ . Note that  $s_n$  does not affect the Vdd-level assignment in  $\mathcal{R}_{n-1}$  even when  $s_n \geq \hat{l}_n$ . Only the upstream switches of  $\hat{b}_n$  are restricted by  $s_n$  while they are also restricted by slacks other than  $s_n$  in  $\mathcal{R}_{n-1}$ . Therefore, those upstream switches have to stay VddH regardless of  $s_n$ . For other switches in  $\mathcal{R}_{n-1}$ , they are not restricted by  $s_n$ , and therefore  $s_n$  does not affect the assignment for them.

Figure 13 shows the situation in which VddL can be assigned to the switch that drives  $\hat{b}_n$  in  $\mathcal{R}_{n-1}$ .  $\mathcal{R}_{n-1}$  is further partitioned into two sub-trees as shown in the figure. We trace back from the switch driving  $\hat{b}_n$  until we reach a VddH switch. The first sub-tree  $\mathcal{R}_B$  is rooted at the immediate downstream switch of that VddH switch while it does not include  $\hat{b}_n$ . As there is no level converter in the tree, VddL has to be assigned to all the switches in  $\mathcal{R}_B$  without considering  $s_n$ . The second sub-tree  $\mathcal{R}_A$  contains all the switches not in  $\hat{b}_n$  and  $\mathcal{R}_B$ .

We first discuss the case in which  $s_n \leq \hat{l}_n$  as shown in Figure 13 (a). VddH has to be assigned to all the upstream switches of  $\hat{b}_n$  in  $\mathcal{R}_B$  considering  $s_n$ . We use  $\hat{b}_n$  be the set of the switches in  $\mathcal{R}_B$  that are assigned to VddH considering  $s_n$  and  $\hat{l}_n$



denote the number of switches in  $\tilde{b}_n$ . The switches in  $\mathcal{R}_B - \tilde{b}_n$  can stay VddL as they are not restricted by  $s_n$ . Hence we have

$$\begin{aligned}
F_n &= F_n|_{\mathcal{R}_A} + F_n|_{\mathcal{R}_B} + F_n|_{\hat{b}_n} \\
&\leq F_{n-1}|_{\mathcal{R}_A} + F_n|_{\mathcal{R}_B - \tilde{b}_n} + F_n|_{\tilde{b}_n} + F_n|_{\hat{b}_n} \\
&\leq N_{n-1}|_{\mathcal{R}_A} + N_n|_{\mathcal{R}_B - \tilde{b}_n} + \frac{s_n}{l_n} \cdot \tilde{l}_n + \frac{s_n}{l_n} \cdot \hat{l}_n \\
&\leq N_{n-1}|_{\mathcal{R}_A} + N_n|_{\mathcal{R}_B - \tilde{b}_n} + s_n \\
&= N_n|_{\mathcal{R}_A} + N_n|_{\mathcal{R}_B - \tilde{b}_n} + N_n|_{\tilde{b}_n} + N_n|_{\hat{b}_n} = N_n
\end{aligned} \tag{21}$$

where  $F_n|_{\mathcal{R}_A}$ ,  $F_n|_{\mathcal{R}_B}$  and  $F_n|_{\hat{b}_n}$  are the estimated VddL switch number in  $\mathcal{R}_A$ ,  $\mathcal{R}_B$ , and  $\hat{b}_n$  considering all the slacks including  $s_n$ .  $F_n|_{\mathcal{R}_A} \leq F_{n-1}|_{\mathcal{R}_A}$  because of (18). We have known  $F_{n-1}|_{\mathcal{R}_A} \leq N_{n-1}|_{\mathcal{R}_A}$  by induction. As VddL is assigned to all the switches in  $\mathcal{R}_B - \tilde{b}_n$ ,  $N_n|_{\mathcal{R}_B - \tilde{b}_n}$  is the number of switches in  $\mathcal{R}_B - \tilde{b}_n$ , i.e.,  $|\mathcal{R}_B - \tilde{b}_n| \cdot F_n|_{\mathcal{R}_B - \tilde{b}_n} \leq |\mathcal{R}_B - \tilde{b}_n|$  as the slack distributed to each switch is smaller than 1. Therefore,  $F_n|_{\mathcal{R}_B - \tilde{b}_n} \leq N_n|_{\mathcal{R}_B - \tilde{b}_n}$ . It is obvious that  $\frac{s_n}{l_n} \cdot \tilde{l}_n + \frac{s_n}{l_n} \cdot \hat{l}_n \leq s_n$  as  $\tilde{l}_n + \hat{l}_n \leq l_n$ .  $N_{n-1}|_{\mathcal{R}_A} = N_n|_{\mathcal{R}_A}$  because  $s_n$  does not affect Vdd-level assignment in  $\mathcal{R}_A$  in this case. The number of VddL switches we can achieve in  $\tilde{b}_n$  and  $\hat{b}_n$  is  $s_n$ , i.e.,  $N_n|_{\tilde{b}_n} + N_n|_{\hat{b}_n} = s_n$ .

Similarly, if  $s_n \geq \hat{l}_n$  as shown in Figure 13 (b), some upstream switches of  $\hat{b}_n$  in  $\mathcal{R}_B$  may have to be assigned to VddH. We have

$$\begin{aligned}
F_n &= F_n|_{\mathcal{R}_A} + F_n|_{\mathcal{R}_B} + F_n|_{\hat{b}_n} \\
&\leq F_{n-1}|_{\mathcal{R}_A} + F_n|_{\mathcal{R}_B - \tilde{b}_n} + F_n|_{\tilde{b}_n} + F_n|_{\hat{b}_n} \\
&\leq N_{n-1}|_{\mathcal{R}_A} + N_n|_{\mathcal{R}_B - \tilde{b}_n} + \frac{s_n}{l_n} \cdot \tilde{l}_n + \frac{s_n}{l_n} \cdot \hat{l}_n \\
&\leq N_{n-1}|_{\mathcal{R}_A} + N_n|_{\mathcal{R}_B - \tilde{b}_n} + \min(s_n, \hat{l}_n + \tilde{l}_n) \\
&= N_n|_{\mathcal{R}_A} + N_n|_{\mathcal{R}_B - \tilde{b}_n} + N_n|_{\tilde{b}_n} + N_n|_{\hat{b}_n} = N_n
\end{aligned} \tag{22}$$

Note that the upstream switches of  $\hat{b}_n$  in  $\mathcal{R}_A$  have to stay VddH as they are restricted by slacks of some sinks other than  $s_n$ . As  $s_n \leq l_n \wedge \tilde{l}_n + \hat{l}_n \leq l_n$ , we have  $\frac{s_n}{l_n} \cdot \tilde{l}_n + \frac{s_n}{l_n} \cdot \hat{l}_n \leq \min(s_n, \hat{l}_n + \tilde{l}_n)$ , where  $\min(s_n, \hat{l}_n + \tilde{l}_n)$  is the number of VddL switches in  $\hat{b}_n$  and the upstream switches of  $\tilde{b}_n$ . Based on (19) ~ (22), we have proved Theorem 1.  $\square$

## SUMMARY OF THE CHANGES

We appreciate all the review comments. After thoroughly considering the questions and concerns raised by all the reviewers, we have made the following changes.

First, we have rerun dTLC-LP for the largest benchmark circuit *clma* with the latest version of linear programming solver [27] and got the result within 16 hours. All the tables and figures have been updated to incorporate the result of benchmark circuit *clma*.

Second, we have implemented two alternative approaches, *dTLC-S-I* and *dTLC-S-U*. Compared to the sensitivity based algorithm dTLC-S that assigns Vdd-level first to logic blocks and then to interconnects, dTLC-S-I assigns Vdd-level first to interconnects and then to logic blocks while dTLC-S-U performs assignment in a uniform fashion to both logic blocks and interconnects. Compared to dTLC-S, dTLC-S-I and dTLC-S-U consume 13.20% and 0.39% more total power, respectively. It is clear that allocating time slack first to logic blocks provides a good starting point for interconnect Vdd-level assignment.

To reflect these changes, we have consistently changed the abstract, introduction and conclusion sections. Below, we will itemize all detailed revisions.

## LIST OF REVISIONS IN DETAIL

## A. Answers to Reviewer #1

*Comment #1 from Reviewer #1*


---

The authors have definitely improved the presentation of the paper. Many errors (technical, grammatical, etc.) in the previous version of the paper have been corrected. Now, I have a much better understanding of the theories and experimental results discussed in the paper.

However, the authors failed to answer some of the critical questions from the reviewers. Thus, I still have quite some concerns about the dTLC-LP algorithm proposed by the authors. As the other reviewer has pointed out, the other algorithms have presented in the preliminary conference version of the paper and therefore the dTLC-LP algorithm should be the major contribution of the paper.

---

Thank you very much for your comments. First, an abstract of all algorithms including dTLC-LP has been presented in conference paper [32]. The reviewer's impression that only dTLC-LP is new in this paper was incorrect. Furthermore, we believe that the primary contribution, as emphasized in the paper, is to propose two problem formulations to avoid using level converters in routing channels. Algorithms to solve the formulations are secondary, as in the general case.

*Comment #2 from Reviewer #1*


---

My first concern is the scalability of the dTLC-LP algorithm. According to the answers from the authors, the performance of the "clma" circuit is not included in Table IV because "dTLC-LP fails to solve the LP problem for the circuit." The authors believe that "the well developed LP solvers in the commercial world ... can solve the LP problem for clma for a much shorter time." Actually, with all the experimental data available, it should not be very difficult to compare the run-time of the publicly available LP solver and that of the commercial ones. At least, I am expecting that the comparison for the "clma" circuit can be shown in the paper if time is limited. Besides, with such scalability concerns for wire segments of the same length, the contribution for extending dTLC-LP to handle wire segments of different lengths is questionable.

---

Thank you very much for your comments and we have made changes to this paper. Please refer to the "Summary of the Changes" at the beginning of this letter.

*Comment #3 from Reviewer #1*


---

My second concern is the performance of the dTLC-LP algorithm. As is stated in the revised version of the paper, "Compared with dTLC-LP, dTLC-S obtains slightly smaller power reduction but run 3X faster". If I am not missing any thing, the "3X" is an average number. Most importantly, even with a "slightly smaller power reduction", the dTLC-S algorithm does not seem to suffer from the scalability problem as the dTLC-LP algorithm and can consistently give out results for all the circuits considered in the paper. Also, judging from Table IV, it seems to me that the dTLC-LP algorithm performs even worse than the dTLC-S algorithm before the refinement step. For example, by checking out column 4 of Table IV, the average "Even with the refinement step, the dTLC-LP algorithm would only achieve 0.52% performance improvement. It is difficult to justify the long run-time suffered by the dTLC-LP algorithm with such a small performance improvement.

Based on the above consideration, I recommend that the authors should "Resubmit after Minor Revision for Review".

---

Thank you very much for you comments. We agree that dTLC-LP without refinement step performs slightly worse compared to dTLC-S. This is mainly due to two reasons as stated at page 9,

"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  $\Delta d$ ). Second, the objective of the LP formulation is to maximize a low bound of power reduction instead of the exact power reduction."

However, the refinement step is part of the dTLC-LP algorithm and the complete dTLC-LP algorithm reduces more power than dTLC-S. Note that dTLC-S cannot further reduce power by using refinement step. As to the efficiency, we have included the following in conclusion at page 13,

“Experiments show that dTLC-LP obtains the lowest power consumption among all the formulations and algorithms. Compared to dTLC-LP, dTLC-S obtains slightly higher power consumption but runs  $3X$  faster. Therefore, it is also an attractive algorithm with a desired tradeoff between runtime and quality.”

Clearly, the comparison between dTLC-LP and dTLC-S enables us to judge the quality of sensitivity based approach.

*Comment #4 from Reviewer #1*

---

A minor presentation issue. The names of the algorithms proposed in the paper appear twice in Table IV.

---

Thank you very much for you comments and we have removed the redundant line in Table IV at page 10.

## B. Answers to Reviewer #2

### *Comment #1 from Reviewer #2*

---

The experimentation section has certainly been improved significantly, and the comparison with previous works is interesting.

I still have a few comments.

What does Fig 8 try to show? I do not understand its significance, and how it answers my original question about how close the lower bound is to the optimal number of Vddl switches.

---

Thank you very much for your comments and we have addressed your questions at page 9 as follows,

“ Figure 8 compares the estimated Vddl switch number given by (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. ”

### *Comment #2 from Reviewer #2*

---

From Theorem 4, it seems that having all 4-length segments improves the performance of the LP algorithm (because the load capacitance of the switches would be approximately equal). My earlier concern that the results may not reflect that of a real FPGA (which consist of different length segments) has not been addressed.

---

Thank you very much for your comments and we have addressed your concern at page 13 as follows,

“ The state-of-the-art commercial FPGAs have used wire segments of different lengths and applied uni-directional level-restore routing switch in routing architecture [29]–[31] to improve performance. Our methodology proposed in this paper can be extended to the interconnects with these novel features. We believe that dTLC formulation can still outperform TLC and h2ILCi 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 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 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. ”

Note that we have shown how to estimate the Vddl switch number with the mix of different wire segment lengths. We do not include it in this paper due to space limit. Neither do we want to defocus this paper.

### *Comment #3 from Reviewer #2*

---

Why are unused level converters not power-gated? Leakage of level converters being 29% of total power is misleading. Since unused logic and interconnect resources are power-gated, it makes sense to power-gate the level converter too.

---

Thank you very much for your comments and we have addressed your concerns at page 0 as follows,

“ 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 to 38.73% of total power and 33.96% of total area respectively for the 20 largest MCNC benchmark circuits [16] at the ITRS 100nm technology node [17]. Power-gating unused level converters may reduce leakage but introduces extra area overhead<sup>7</sup>. Therefore it is less attractive compared to the methods in this paper that remove level converters. ”

### *Comment #4 from Reviewer #2*

---

On page 2, “Compared to [13], the best algorithm ... power by 64.06%”. Does this mean that the power reduction is 64.06% of that in [13]? Similar sentences have been used at several places in the paper (results shown compared to [13]).

---

Thank you very much for your comments and we have addressed your concern at page 1 and 13 as follows,

<sup>7</sup>Our experiments show that using a 1X PMOS transistor as a power transistor, power-gating can reduce leakage power of an unused level converter by 1000X with 8% level converter delay increase and 3% chip-level area overhead at the ITRS 100nm technology node.

“ The best algorithm, dTLC-LP, reduces total interconnect power by 52.90% compared to the *segment based level converter insertion (SLC)* [13], and by 18.52% compared to the best approach proposed in [12]. ”

“ Without loss of performance, the best algorithm dTLC-LP reduces total interconnect power by 52.90% compared to the *segment based level converter insertion (SLC)* proposed in [13], and by 18.52% compared to the best approach *h2ILCi* proposed in [12] where all routing trees driven by one CLB have the same Vdd-level as the source CLB. ”

*Comment #5 from Reviewer #2*

---

Some typo in the last line of Sec 5A first para. ” ... does not allow the interface of different Vdd levels with a routing tree ...”

---

Thank you very much for your comments. We have corrected the typo at page 9 as follows,

“ This is because that TLC uses a routing tree as the assignment unit and does not allow the mix of different Vdd-levels within a routing tree. ”

*Comment #1 from Reviewer #3*

---

Using Dual-Vdd in FPGAs is a well known concept, and the authors' idea to eliminate the need of level converters is interesting. Three heuristics for power efficient dual-Vdd assignment for FPGA devices is presented in the paper. As power is a major design concern today, the techniques presented in this paper are interesting. Though the ideas in the paper are not novel, but the authors do a good job in implementing their algorithms to eliminate the need of level converters. But, I would strongly recommend that the authors make a better presentation and explain ideas in a clear and concise manner. Here are my comments:

1. If the logic blocks are assigned a Vdd level based on sensitivity analysis, and after that when the switch blocks are assigned corresponding Vdd levels, are the authors taking care of the fact that assigning VddL to a logic block may increase the delay, and correspondingly the switch/connection blocks will have less slack now? It is not very clear.

---

Thank you very much for your comments and we have addressed your concerns in the footnote at page 9 as follows, "All the algorithms SLC, TLC-S, dTLC-S and dTLC-LP do consider the fact that VddL logic blocks consume time slack."

*Comment #2 from Reviewer #3*

---

2. When a logic block is assigned VddH, a VddL track cannot drive it. The authors only discuss that the need for level converters is eliminated because a VddL track never drives a VddH track. What if a VddL wire is connected to the input of a VddH logic block? The authors should elaborate on how they achieve this, if at all they are taking care of this fact.

---

Thank you very much for your comments and we have addressed your concerns at page 1 as follows,

"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."

*Comment #3 from Reviewer #3*

---

3. Will assigning Vdd levels to logic blocks before assigning Vdd levels to tracks affect the quality of the results? Interconnect delay is the major component in delay compared to logic blocks, so will assigning Vdd levels to the tracks first yield better results? The authors should justify why they are assigning Vdd levels to logic blocks first.

---

Thank you very much for your comments and we have made changes to the paper. Please refer to the "Summary of the Changes" at the beginning of the letter. More details can be found in Section V-B at page 10 as follows,

"In this paper, we first perform a sensitivity based Vdd-level assignment for Vdd-programmable logic blocks before assigning 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 algorithm dTLC-S, dTLC-S-I and dTLC-S-U, all implemented based on power sensitivity. Column 2-7 in the Table 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 percentage of VddL CLBs and switches. Column 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."