revision under review by
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

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 power sensitivity based algorithms, *TLC-S* and *dTLC-S*, with implicit time slack allocation and a linear programming (LP) based algorithm, *dTLC-LP*, with explicit time slack allocation. All first allocate time slack to interconnects with higher power sensitivity and assign low-Vdd to them for more power reduction. Compared to the aforementioned Vdd-programmable method using Vdd-level converters, the best algorithm dTLC-LP reduces interconnect power by 64.06% without performance loss for the MCNC benchmark circuits. Compared to dTLC-LP, dTLC-S obtains slightly smaller power reduction but runs $3X$ faster.

Index Terms

Power_minimization, Low-power_design, Interconnect, Optimization.

I. Introduction

FPGA power modeling and reduction has become an active research 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 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. The segment based level converter insertion, called as *SLC*, provides fine-grained Vdd-programmability for interconnects and may help to maximize dynamic power reduction. However, SLC also introduces large leakage overhead. As presented in Table I, the average leakage due to level converters in routing channels is 29% of total power for the 20 largest MCNC benchmark circuits [15] at the ITRS $100nm$ technology node [16] with power-gating [1] of unused interconnect switches [13], [17].

There are some previous approaches to avoid directly using level converters in Vdd-programmable interconnects. [14] uses level-restore buffers to avoid using level converters in interconnects. NMOS power transistor is used to generate the VddL level leveraging the threshold voltage drop of NMOS transistor when passing '1'. However, the range of VddL is limited by the threshold voltage of NMOS power transistor. The level-restore PMOS transistor 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 used in [13]. [12] 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, the assignment is lacking of flexibility and cannot effectively leverage the surplus timing slack existing in FPGA designs.

[1] Power-gating unused level converters may reduce leakage, but is less attractive compared to methods in this paper that remove level converters.

circuit	total power (watt)	# of LCs	leakage of LCs	
			power (watt)	% of total power
alu4	0.10956	122980	0.01771	16.16%
apex2	0.13507	201536	0.02902	21.49%
apex4	0.07986	137020	0.01973	24.71%
bigkey	0.22706	285740	0.04115	18.12%
clma	0.74067	2050655	0.29529	39.87%
des	0.25343	415715	0.05986	23.62%
diffeq	0.05062	128921	0.01856	36.67%
dsip	0.24545	357175	0.05143	20.95%
elliptic	0.16033	438283	0.06311	39.37%
ex1010	0.23347	584995	0.08424	36.08%
ex5p	0.07924	143065	0.02060	26.00%
frisc	0.29861	944892	0.13606	45.57%
misex3	0.09974	132440	0.01907	19.12%
pdc	0.34530	878867	0.12656	36.65%
s298	0.07383	157652	0.02270	30.75%
s38417	0.34903	743341	0.10704	30.67%
s38584	0.30223	602095	0.08670	28.69%
seq	0.13901	201536	0.02902	20.88%
spla	0.21521	498311	0.07176	33.34%
tseng	0.04075	88660	0.01277	31.33%
avg.				29.00%

TABLE I

LEAKAGE POWER OF VDD-LEVEL CONVERTERS IN VDD-PROGRAMMABLE INTERCONNECTS WITH SEGMENT BASED LEVEL-CONVERTER INSERTION USING $100nm$ TECHNOLOGY. (LC STANDS FOR LEVEL CONVERTER.)

In this paper, we use the Vdd-programmable interconnects from [13], but remove the level converters in routing channels by developing novel Vdd-level assignment algorithms to guarantee that no VddL switch drives VddH switches. Our first contribution is that we propose two ways to avoid using level converters in interconnects. Same as [13], level converters are inserted at CLB inputs and outputs, and can be used when needed. In the first approach, we enforce that there is only one Vdd-level within each routing tree, namely, *tree based level converter insertion (TLC)*. In the second approach, we can have different Vdd-levels within a routing tree, but no VddL switch drives VddH switches, namely, *dual-Vdd tree based level converter insertion (dTLC)*.

Our second contribution is that we 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 and minimize the chip-level 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. Compared to [13], the best algorithm dTLC-LP reduces total interconnect power by 64.06% without performance loss. In contrast, the best approach proposed in [12] achieves 55.45% power reduction. Compared to dTLC-LP, dTLC-S obtains slightly smaller power reduction but runs $3X$ faster.

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 [17] [2].

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. We use CLBs with cluster size $N = 10$ and LUT size $k = 4$ in our study. 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

[2]However, the emphasis of [17] is architecture evaluation considering Vdd programmability.

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 mix of wire segments of different lengths in the future.

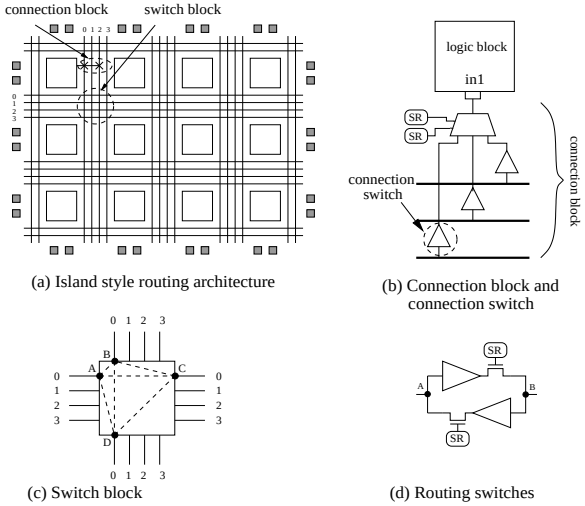

Fig. 1. (a) Island style routing architecture; (b) Connection block; (c) Switch block; (d) Routing switches. (SR stands for SRAM cell.)

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.

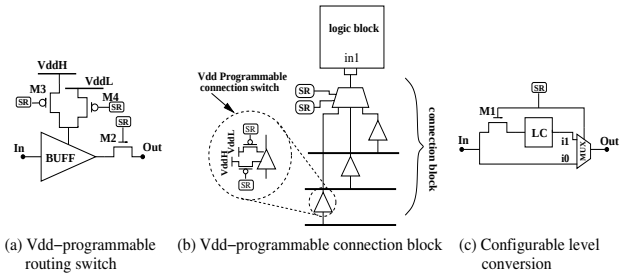

Fig. 2. (a) Vdd-programmable routing switch; (b) Vdd-programmable connection block; (c) Configurable Vdd-level conversion. (SR stands for SRAM cell and LC stands for level converter.)

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]. Same as [13], in this paper 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. However, the segment based level converter insertion, called as *SLC* in this paper, introduces large leakage overhead. Analysis (see Table I) shows that the average leakage of the level converters in routing channels is 29% of total power 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. Same as [13], configurable level converters are inserted at CLB inputs and outputs, and can be used when needed. 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, we can have different Vdd-levels 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 that VddH switch drives VddL switches within one routing tree for dTLC. To make the presentation simple, we

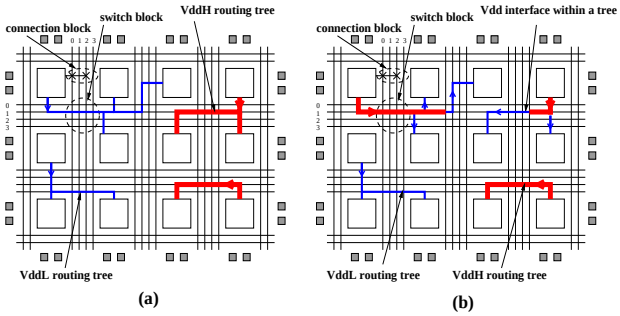

Fig. 3. (a) Tree based level converter insertion; (b) Dual-Vdd tree based level converter insertion.

summarize the notations frequently used in this paper in Table II. They will be explained in detail when first used.

$\mathcal{G}(\mathcal{V},\mathcal{E})$	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 INPUT AND LOAD CAPACITANCE OF A 7X BUFFER CALIBRATED 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 power-gated unused switches 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,

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

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 easily 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,

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

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

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

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

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

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

i.e., no VddL switch should drive VddH switches. $\mathcal{FO}_{ij}$ gives the set of fanout switches of j^{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. Alternatively, 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

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

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

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

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

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

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

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 $\Delta P/\Delta V_{dd}$, which is defined as the power reduction (ΔP) divided by the Vdd-level difference (ΔVdd) between VddH and VddL, for each switch with the wire it drives. 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 $\sum_{j=0}^{N_s(i)-1} \Delta P_{ij}/\Delta V_{dd}$, where $\Delta P_{ij}/\Delta V_{dd}$ is the power sensitivity of j^{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(∃ '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 for dTLC problem. The details of these two algorithms are presented as 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. We define a switch as a *candidate switch* if it is 'untried', and it does not drive any switch or VddL has been assigned to all of its fanout switches. 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 violation, 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 summarized in Figure 5.

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

Fig. 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 for dTLC problem. 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} in a multiple of Δd, where Δd 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. $S0$ and $S1$ are the slacks allocated to two sinks $Sink0$ and $Sink1$, respectively. In Figure 6 (a), VddL can be assigned to $b2$ given $S0 = 1$ and VddL can be assigned to $b3$ given $S1 = 1$. When we increase the slack $S1$ for $Sink1$ to 2 in Figure 6 (b), $b0$ has to stay VddH restricted by $S0 = 1$. In other words, $b0$ is restricted by both $S0$ and $S1$, and VddL can only be assigned to $b0$ when $S0 \geq 3 \wedge S1 \geq 2$. Figure 6 (c) shows the case in which VddL is assigned to all the switches given $S0 = 3 \wedge S1 = 2$. Therefore, there is an upper bound for slack, which is the delay increase when VddL are assigned to all the switches in a tree, and 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.

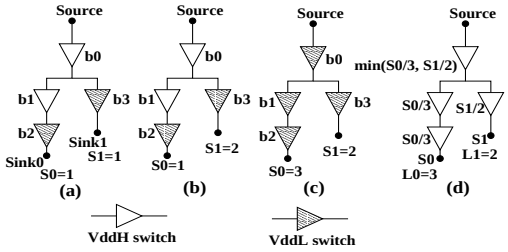

Fig. 6. An example for estimating number of VddL switches.

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 that can be

achieved given the allocated slack as

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

To estimate the number of VddL switches that can be achieved 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 in a multiple of Δd, (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

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

s.t.

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

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

The slack s_{ik} is a continuous variable normalized to Δd in (12) and also the rest part of the paper. The floor function $\lfloor s_{ik} \rfloor$ in (12) gives multiple of Δd 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,

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

The slack upper bound constraints can be expressed as

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

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

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

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

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

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

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

We formulate the *time slack allocation problem* using objective function (10), additional constraints (11) and (13), slack upper bound constraints (14), plus timing constraints (5), (6), (15) and (16). It is easy to verify that (5), (6), (11) and (13) $\sim$ (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 for each routing tree 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 R_i:
Assign VddH to all switches in R_i and mark them as 'untried';
While( ∃ candidate switch)
{
    Assign VddL to a candidate switch j;
    If (timing constraints violated)
    {
        Find all the upstream switches of j in R_i;
        Assign VddH to j and those upstream switches, and
mark them as 'tried';
    }
    Else mark j as 'tried';
}
```

Fig. 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 that is better than the optimal solution. Therefore, the bottom-up assignment algorithm gives the optimal solution when level converters cannot be used. □

Theorem 4: Given a routing tree $\mathcal{R}_i$ in which each switch has a uniform load capacitance, and transition density of the routing tree [3], and 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 get another solution with 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. □

- **Refinement**

After net-level assignment, we may further reduce power by leveraging surplus slack. Figure 6(b) shows a routing tree containing surplus slack. $b0$ has to stay VddH restricted by $S0 = 1$. Therefore, $Sink1$ can only consume one unit slack from $S1$ 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*, which are proposed in this paper. We then compare the best one among TLC-S, dTLC-S and dTLC-LP, and one previous approach without using level converters in interconnects, *h2lLCi* [12], to the baseline using Vdd-programmable interconnects with fine-grained Vdd-level converter inserted in routing channels, *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 $\sim$ 0.7 suggested in [28], we use 1.3v for VddH and 0.8v for VddL in our experiments at $100nm$ technology node.

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

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 logic blocks without performance loss. One 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* [17] is then used to calculate power. It has been shown that fpgaEva-LP2 achieves high fidelity as well as high accuracy compared to SPICE simulation with the average of absolute error 8.26% [17]. 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 are presented in the power difference normalized to TLC-S in this section. The benchmark *clma* is not presented in the table as dTLC-LP fails to solve the LP problem for the circuit. 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 56.05%, 77.02% and 77.54% 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 interface of different Vdd-levels with 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 13.07% and 14.87% 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 13.87% and 15.76% less dynamic power, 6.73% and 6.90% less leakage power, respectivly. 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 number of VddL switches achieved by dTLC-LP before refinement and the estimated VddL switch number given by (11). It is clear that (11) consistently gives a lower bound of VddL switch number that can be achieved. The average error due to estimation is 4.68%.

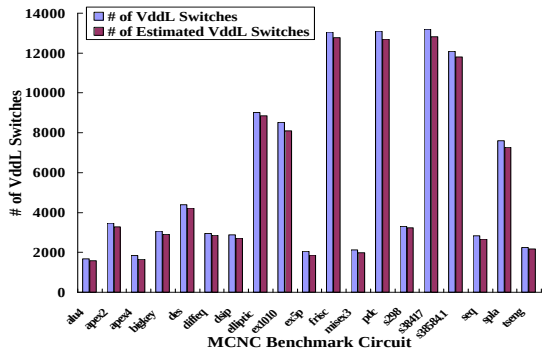

Fig. 8. Comparison between VddL switch number achieved by dTLC-LP before refinement and the estimated VddL switch number.

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 3.71% VddL switches. Compared to TLC-S, the refinement step obtains 2.25% interconnect power reduction, 2.65% interconnect dynamic power reduction and 1.17% 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.

2) Runtime Comparison: Table V compares the runtime [4] between the three algorithms. TLC-S is the fastest among the three algorithms. dTLC-S and dTLC-LP take $1.64X$ and $5.33X$ 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. For the largest circuit *clma*, dTLC-LP fails to solve the LP problem after running 30 hours. We believe that the well developed linear programming solvers in the commercial world such as [26] can solve the LP problem for *clma* in a much shorter time. 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.

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

1	2	3	4	5	6	7	8	9	10	11	12	13
circuit	% of VddL switches			interconnect power			interconnect dynamic power			interconnect leakage power		
	TLC-S	dTLC-S	dTLC-LP (due to refinement)	TLC-S (watt)	dTLC-S	dTLC-LP (due to refinement)	TLC-S (watt)	dTLC-S	dTLC-LP (due to refinement)	TLC-S (watt)	dTLC-S	dTLC-LP (due to refinement)
Circuit	TLC-S	dTLC-S	dTLC-LP	TLC-S	dTLC-S	dTLC-LP	TLC-S	dTLC-S	dTLC-LP	TLC-S	dTLC-S	dTLC-LP
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%)
des	60.03%	79.90%	80.38% (2.68%)	0.10906	-11.40%	-13.44% (-1.83%)	0.10266	-11.82%	-13.97% (-1.91%)	0.00641	-4.80%	-4.97% (-0.57%)
diffeq	80.52%	91.23%	91.56% (3.22%)	0.00771	-2.47%	-2.17% (-4.72%)	0.00467	-1.85%	-1.28% (-7.14%)	0.00304	-3.43%	-3.54% (-1.00%)
dsip	61.01%	77.69%	78.01% (1.28%)	0.10546	-12.80%	-12.91% (-1.09%)	0.10023	-13.29%	-13.41% (-1.13%)	0.00524	-3.36%	-3.43% (-0.27%)
elliptic	79.46%	93.69%	94.09% (1.51%)	0.02483	-3.01%	-3.46% (-1.32%)	0.01685	-2.05%	-2.64% (-1.70%)	0.00798	-5.05%	-5.19% (-0.53%)
ex1010	43.93%	69.21%	69.66% (5.99%)	0.06501	-20.05%	-22.93% (-1.37%)	0.05336	-22.59%	-26.06% (-1.23%)	0.01164	-8.45%	-8.60% (-2.00%)
ex5p	38.12%	65.50%	64.45% (4.72%)	0.02818	-14.68%	-15.09% (-1.29%)	0.02512	-15.40%	-15.89% (-1.27%)	0.00306	-8.84%	-8.49% (-1.48%)
frisc	95.80%	99.06%	99.07% (8.99%)	0.02431	-1.25%	-1.34% (-5.00%)	0.01167	-1.43%	-1.61% (-7.16%)	0.01264	-1.08%	-1.08% (-3.00%)
misex3	37.12%	64.58%	65.90% (3.26%)	0.05168	-14.90%	-18.38% (-1.56%)	0.04869	-15.25%	-18.92% (-1.59%)	0.00298	-9.10%	-9.56% (-1.08%)
pdc	37.65%	71.34%	72.24% (3.79%)	0.09887	-23.32%	-26.44% (-1.74%)	0.08346	-25.38%	-29.01% (-1.82%)	0.01541	-12.17%	-12.50% (-1.35%)
s298	39.18%	81.28%	81.88% (2.00%)	0.02445	-28.76%	-31.04% (-0.39%)	0.02090	-31.21%	-33.84% (-0.34%)	0.00355	-14.30%	-14.50% (-0.68%)
s38417	75.16%	85.40%	86.31% (3.47%)	0.09803	-6.58%	-9.88% (-3.43%)	0.08295	-7.20%	-11.05% (-3.86%)	0.01508	-3.15%	-3.43% (-1.04%)
s38584	87.56%	94.42%	94.56% (3.57%)	0.08829	-5.49%	-6.74% (-1.56%)	0.07637	-6.00%	-7.44% (-1.65%)	0.01192	-2.20%	-2.26% (-1.02%)
seq	33.04%	61.38%	62.21% (3.38%)	0.07198	-18.36%	-22.21% (-2.20%)	0.06765	-18.95%	-23.03% (-2.28%)	0.00434	-9.13%	-9.43% (-1.07%)
spla	32.08%	69.35%	70.69% (4.02%)	0.07343	-24.89%	-27.87% (-2.24%)	0.06377	-26.72%	-30.08% (-2.37%)	0.00966	-12.79%	-13.26% (-1.34%)
tseng	93.34%	95.88%	96.55% (3.33%)	0.00850	-1.27%	0.22% (-6.49%)	0.00643	-1.40%	0.64% (-8.26%)	0.00207	-0.86%	-1.08% (-1.00%)
avg.	56.05%	77.02%	77.54% (3.71%)	-	-13.07%	-14.87% (-2.25%)		-13.78%	-15.76% (-2.65%)	-	-6.73%	-6.90% (-1.17%)

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	>30H
des	15243	176.65	287.81	1054.74
diffeq	13664	113.81	143.95	553.84
dsip	11444	96.45	131.62	406.96
elliptic	30192	607.85	913.04	3136.59
ex1010	33265	836.32	1422.79	5109.22
ex5p	8722	62.11	93.99	187.44
frisc	40662	1135.84	1912.15	6135.38
misex3	10271	74.35	106.72	276.41
pdc	40001	1254.57	2508.57	8210.07
s298	16852	179.72	238.18	837.22
s38417	57503	1821.09	2895.79	9152.52
s38584	46014	1255.31	1892.86	6863.62
seq	13426	129.22	203.01	509.22
spla	27908	524.76	1009.07	3339.51
tseng	9603	52.45	71.53	163.55
geometric mean		$1X$	$1.64X$	$5.33X$

TABLE V

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

B. Comparison Between SLC, h2lLCi and dTLC-LP with Relaxed Timing Specification

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

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, h2lLCi and dTLC-LP. The VddL switch percentage and critical path delay are the arithmetic and geometric mean over the MCNC benchmark set, respectively. When the timing specification is not relaxed, SLC, h2lLCi and dTLC-LP achieve 74.70%, 41.80% and 77.54%, respectively. Both of SLC and dTLC-LP achieve more VddL switches than h2lLCi. It is because that all the trees driven by one CLB have the same Vdd-level as the source CLB and VddH circuit element is not allowed to drive VddL circuit element without the presence of level converter in h2lLCi. dTLC-LP consistently achieves the highest percentage of VddL switches compared to previous approach SLC [5] and h2lLCi at different relaxed delays.

Figure 10 compares the interconnect power and critical path delay tradeoff curves achieved by SLC, h2lLCi and dTLC-LP. Both the interconnect power and critical path delay are the geometric mean over the MCNC benchmark set. Compared to SLC,

[5]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.

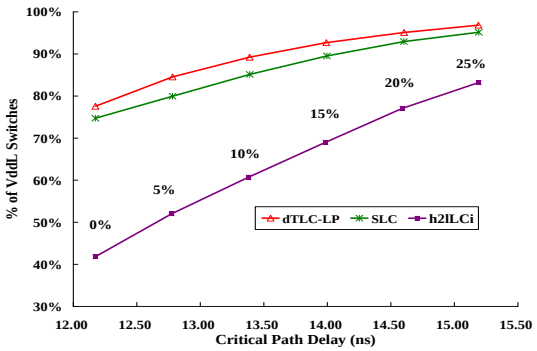

Fig. 9. Percentage of VddL interconnect switches vs. critical path delay curves for SLC, h2lLCi and dTLC-LP.

h2lLCi and dTLC-LP reduce interconnect power by 55.45% and 64.06% without relaxing timing specification, respectively. dTLC-LP consistently achieves the lowest power compared to SLC and h2lLCi at different relaxed delays. Compared to SLC at the same relaxed delay, the LP based algorithm achieves 89.22% VddL switches and reduces interconnect power by 69.05% when we relax critical path delay by 10%. The power gap between dTLC-LP and h2lLCi decreases at larger relaxed delay. 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.

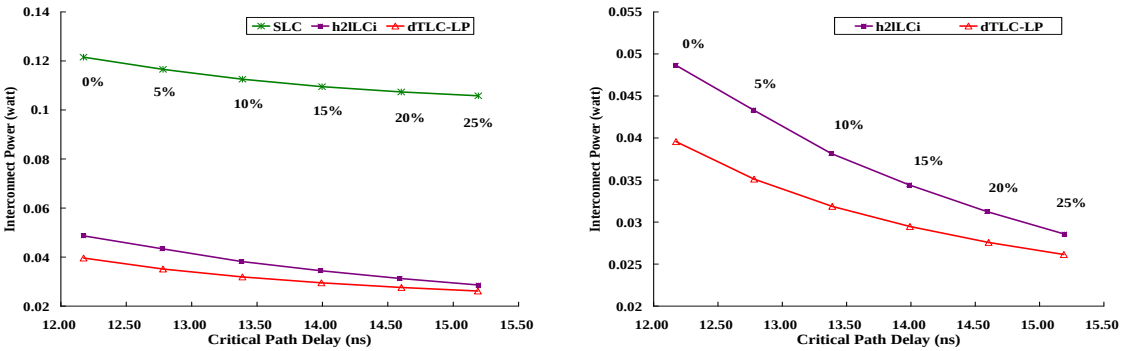

Fig. 10. Interconnect power and critical path delay tradeoff for SLC, h2lLCi and dTLC-LP.

2) Area Comparison: Table VI present the FPGA total area given by single-Vdd, SLC, h2lLCi and dTLC-LP, respectively. We use the same area model from [22], in which area is counted in number of minimum width transistor areas 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:

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

As presented in the table, the area overhead given by SLC, h2lLCi and dTLC-LP are 151.33%, 65.02% and 65.53% compared to the single-Vdd FPGA, respectively. h2lLCi and dTLC-LP have almost the same area, and TLC-S, dTLC-S and dTLC-LP have the same area as all of them use the same Vdd-programmable interconnects with level converters inserted at CLB inputs and outputs.

VI. Conclusions

Considering the need to remove Vdd-level converters in routing channels that introduce large leakage overhead, we have proposed 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)*. In the second approach, we can have different Vdd-levels within a routing tree, but no VddL switch drives VddH switches, namely, *dual-Vdd tree based level converter insertion (dTLC)*.

We propose 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

circuit	single-Vdd	SLC		h2lLCi		dTLC-LP	
	area	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%
diffeq	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%
pdc	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 VI

FPGA AREA COMPARISON BETWEEN SINGLE-VDD, SLC, H2LLCI 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.

allocate time slack first to interconnects with higher 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. Compared to [13], the best algorithm dTLC-LP reduces total interconnect power by 64.06% without performance loss. In contrast, the best approach *h2lLCi* proposed in [12] achieves 55.45% power reduction. Compared to dTLC-LP, dTLC-S obtains slightly smaller power reduction but runs $3X$ faster.

The state-of-art commercial FPGAs have applied uni-directional level-restore routing switch in routing architecture [29]–[31]. Our methodology proposed in this paper can be directly used for the interconnects with these novel features. We believe that dTLC formulation can still outperform TLC and h2lLCi as dTLC allows interface of different Vdd-levels within a routing tree and is able to effectively leverage the time slack existing in FPGA designs, which can be verified as the future work.

In the future, we will study simultaneous Vdd-level assignment for logic block and interconnects. The algorithms proposed in this paper allocate time slack first to logic blocks followed by interconnects. Allocating time slack to both logic blocks and interconnects in a uniform fashion may reduce more power. 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. We used computers donated by Intel. Address comments to lhe@ee.ucla.edu. The authors like to thank Mr. 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*, Februray 2004.
[6] S. Srinivasan, A. Gayasen, and T. Tuan, "Leakage control in fpga routing fabric," in *Proc. Asia South Pacifi c 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*, Februray 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*, Februray 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 Pacifi c 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] S. Yang, "Logic synthesis and optimization benchmarks, version 3.0," Microelectronics Center of North Carolina (MCNC), Tech. Rep., 1991.

[16] International Technology Roadmap for Semiconductor, in *http://public.itrs.net/*, 2003.

[17] 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*, Februray 2005.

[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 3.2: a public domain (MI)LP solver*. 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*.

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 $\mathcal{R}$.

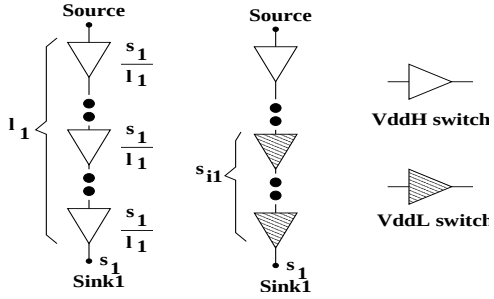

Fig. 11. A 1-sink routing tree $\mathcal{R}$.

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 lost of generality, we arrange the sink order such that for n^{th} sink,

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

We trace back $\mathcal{R}_n$ from n^{th} sink and find the first branching point. Let $\widehat{b}_n$ be the branch from the immediate downstream switch of the branching point to n^{th} sink, and $\widehat{l}_n$ denote the number of switches in $\widehat{b}_n$. Note that the switches in $\widehat{b}_n$ are restricted only by s_n. Suppose we remove the branch $\widehat{b}_n$ 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 $\widehat{b}_n$ 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 $\widehat{b}_n$ in $\mathcal{R}_{n-1}$. As there is no level converter in $\mathcal{R}_n$, all the upstream switches of $\widehat{b}_n$ have to stay VddH. Note that VddL cannot be assigned to the upstream switches of $\widehat{b}_n$ 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

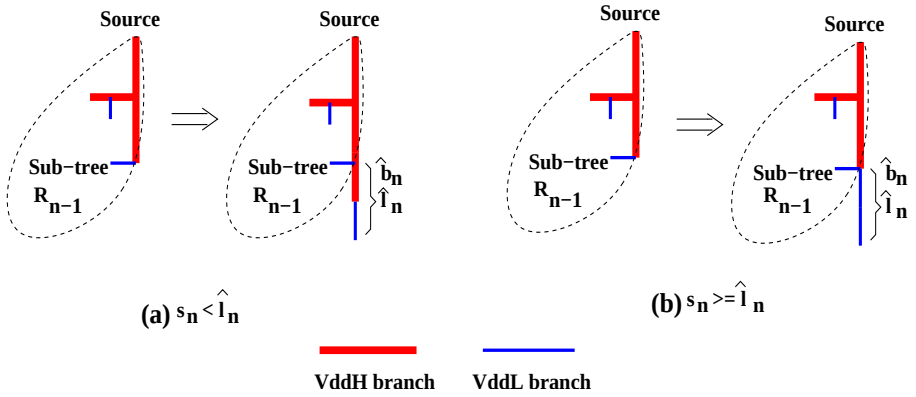

Fig. 12. A n-sink routing tree $\mathcal{R}_n$ in which VddL cannot be assigned to the switch that drives $\widehat{b}_n$ in $\mathcal{R}_{n-1}$.

estimated number of VddL switches that can be achieved in the subtree $\mathcal{R}_{n-1}$, respectively. If $s_n < \widehat{l}_n$ (see Figure 12(a)), we can assign VddL to the bottom s_n switches in $\widehat{b}_n$, and we have

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

where $F_n|_{\mathcal{R}_{n-1}}$ and $F_n|_{\widehat{b}_n}$ are the estimated VddL switch number in $\mathcal{R}_{n-1}$ and $\widehat{b}_n$ 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 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 $s_n \leq \widehat{l}_n$ and therefore s_n does not affect the Vdd-level assignment in $\mathcal{R}_{n-1}$. It is clear that $\frac{s_n}{l_n} \cdot \widehat{l}_n \leq s_n$ as $\widehat{l}_n \leq l_n$.

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

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

$\frac{s_n}{l_n} \cdot \widehat{l}_n \leq \widehat{l}_n$ 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 $s_n \geq \widehat{l}_n$. Only the upstream switches of $\widehat{b}_n$ 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 $\widehat{b}_n$ 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 $\widehat{b}_n$ until we reach a VddH switch.

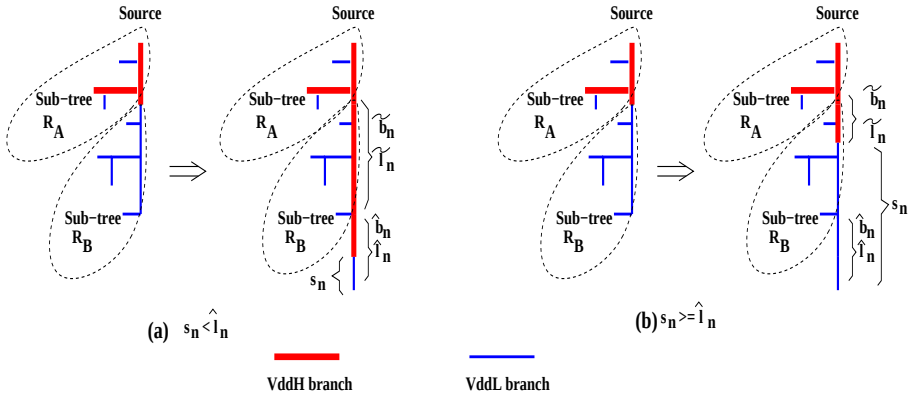

Fig. 13. A n-sink routing tree $\mathcal{R}_n$ in which VddL can be assigned to the switch that drives $\widehat{b}_n$ in $\mathcal{R}_{n-1}$.

The first sub-tree $\mathcal{R}_B$ is rooted at the immediate downstream switch of that VddH switch while it does not include $\widehat{b}_n$. 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 $\widehat{b}_n$ and $\mathcal{R}_B$.

We first discuss the case in which $s_n \leq \widehat{l}_n$ as shown in Figure 13 (a). VddH has to be assigned to all the upstream switches of $\widehat{b}_n$ in $\mathcal{R}_B$ considering s_n. We use $\widetilde{b}_n$ be the set of the switches in $\mathcal{R}_B$ that are assigned to VddH considering s_n and $\widetilde{l}_n$ denote the number of switches in $\widetilde{b}_n$. The switches in $\mathcal{R}_B - \widetilde{b}_n$ can stay VddL as they are not restricted by s_n. Hence we have

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

where $F_n|_{\mathcal{R}_A}$, $F_n|_{\mathcal{R}_B}$ and $F_n|_{\widehat{b}_n}$ are the estimated VddL switch number in $\mathcal{R}_A$, $\mathcal{R}_B$, and $\widehat{b}_n$ 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 $\mathcal{R}_B - \widetilde{b}_n$, $N_n|_{\mathcal{R}_B - \widetilde{b}_n}$ is the number of switches in $\mathcal{R}_B - \widetilde{b}_n$, i.e., $|\mathcal{R}_B - \widetilde{b}_n|$. $F_n|_{\mathcal{R}_B - \widetilde{b}_n} \leq |\mathcal{R}_B - \widetilde{b}_n|$ as the slack distributed to each switch is smaller than 1. Therefore, $F_n|_{\mathcal{R}_B - \widetilde{b}_n} \leq N_n|_{\mathcal{R}_B - \widetilde{b}_n}$. It is obvious that $\frac{s_n}{l_n} \cdot \widetilde{l}_n + \frac{s_n}{l_n} \cdot \widehat{l}_n \leq s_n$ as $\widetilde{l}_n + \widehat{l}_n \leq l_n$. $N_{n-1}|_{\mathcal{R}_A} = N_n|_{\mathcal{R}_A}$ 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 $\widetilde{b}_n$ and $\widehat{b}_n$ is s_n, i.e., $N_n|_{\widetilde{b}_n} + N_n|_{\widehat{b}_n} = s_n$.

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

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

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