PAPER UNDER REVIEW, PLS DO NOT DISTRIBUTE Leakage Efficient Chip-Level Dual-Vdd Assignment with Time Slack Allocation for FPGA Power Reduction

Yan Lin and Lei He
Electrical Engineering Department
University of California, Los Angeles
*

ABSTRACT

To reduce 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 Vdd-programmable method consume a large amount of leakage. In this paper, we develop chip-level dual-Vdd assignment algorithms to guarantee that no low-Vdd interconnect switch drives high-Vdd interconnect switches. This removes the need of Vdd-level converters and reduces interconnect leakage and interconnect device area by 91.78% and 25.48%, respectively. The assignment algorithms include power sensitivity based heuristics with implicit time slack allocation and a linear programming (LP) based method with explicit time slack allocation. Both first allocate time slack to interconnects with higher transition density and assign low-Vdd to them for more power reduction. Compared to the aforementioned Vdd-programmable method using Vdd-level converters, the LP based algorithm reduces interconnect power by 65.13% without performance loss for the MCNC benchmark circuits. Compared to the LP based algorithm, the sensitivity based heuristics can obtain slightly smaller power reduction but run $4X$ faster.

1. 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] proposes configuration inversion method to reduce leakage power of multiplexers without any additional hardware. [4] studies a suite of power-aware FPGA CAD algorithms without changing the existing FPGAs. In addition, low power FPGA circuits and architectures have been proposed, including power-gating unused FPGA logic blocks [5] to reduce leakage power, and Vdd-programmability for FPGA logic blocks [6, 7] and interconnects [8, 9] to reduce both dynamic and leakage power.

In this paper, we are aiming at improving the Vdd-programmable interconnects proposed in [8], 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. However, the fine-grained level converter insertion introduces large leakage overhead. Analysis shows that the leakage overhead of level converters in routing channels is 36% of total power for circuit $s38584$. In this paper, we use the Vdd-programmable interconnects same as that in [8], but remove the level converters in routing channels by developing novel CAD algorithms. Experimental results show that compared to [8], we reduce interconnect leakage power and device area by 91.78% and 25.48%, respectively, and reduce total interconnect power by up to 65.13% without performance loss.

The rest part of the paper is organized as follows. Section 2 presents modeling and problem formulation. Section 3 presents both power sensitivity based heuristics and a linear programming (LP) based algorithm. Section 4 presents the experimental results and Section 5 concludes this paper.

2. MODELING AND PROBLEM FORMULATION

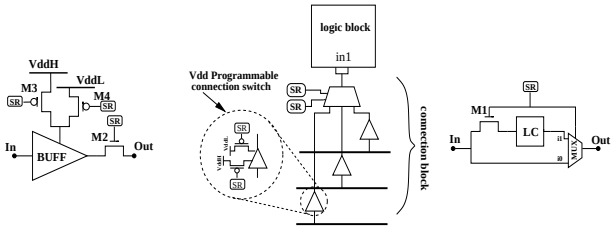
2.1 Preliminaries

Interconnects consume most of the area and power of FPGAs. We assume the traditional island style routing architecture in our study. The logic blocks are surrounded by routing channels consisting of wire segments. The input and output pins of a logic block can be connected to the wire segments in the surrounding channels via a *connection switch*. Wire segments can be formed into a long connection via a *routing switch* at each intersection of a horizontal channel and a vertical channel. Routing switches and connection switches can be implemented by tri-state buffers and buffers, respectively. An *interconnect switch* is either a routing switch or a connection switch. An *interconnect segment* is a wire segment driven by an interconnect switch. For simplicity, we assume a uniform length 4 for all wire segments.

Vdd programmability can be applied to interconnects to reduce FPGA power. Figure 1 shows the Vdd-programmable interconnect switch proposed in [8]. For the Vdd-programmable routing switch in Figure 1 (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. 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. Another type of routing resources is the connection block in Figure 1 (b). The multiplexer-based implementation chooses only one track in the routing channel and connects it to the logic block input pin. The buffers between the routing tracks and the multiplexer are connection switches. Similar to the routing switch, programmable-Vdd is

*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.

also applied to the connection switch.



(a) Vdd−programmable routing switch (b) Vdd−programmable connection block (c) Configurable level conversion

Figure 1: (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 1 (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 [8]. Same as [8], 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.

2.2 Motivation

The fine-grained Vdd-level converter insertion introduces large leakage overhead. Analysis shows that the leakage overhead of the level converters in routing channels is 36% of total power for circuit $s38584$. *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[1]. In the first approach, we enforce that there is only one Vdd-level within each routing tree, namely, *tree based assignment*. In the second approach, we can have different Vdd-levels within a routing tree, but no VddL switch drives VddH switches, namely, *segment based assignment*. To make the presentation simple, we summarize the notations frequently used in this paper in Table 1. They will be explained in detail when first used.

2.3 Delay Modeling with Dual-Vdd

A directed acyclic timing graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ [10] 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 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. Vdd-level has little impact on the input and load capacitance of a switch, and such impact is ignored in this paper.

[1]Same as [8], configurable level converters are inserted at logic block inputs and outputs, and can be used when needed.

$\mathcal{G}(\mathcal{V}, \mathcal{E})$	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 1: Notations frequently used in this paper.

2.4 Power Modeling with Dual-Vdd

There are three power sources in FPGAs, switching power, short-circuit power and static 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, static 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 4.

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

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

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. For simplicity, we assume that all the switches have the same load capacitance and apply the constant leakage scaling proposed in [6], i.e., the leakage power of an interconnect switch is same under different Vdd-levels. Our algorithms can however be easily extended to remove these simplifications. v_{ij} indicates Vdd-level of j^{th} switch in $\mathcal{R}_i$ as follows

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

The interconnect power reduction P_r using programmable dual-

Vdd can be expressed as

$$P_r \propto \sum_{i=0}^{N_r-1} f_s(i) \sum_{j=0}^{N_s(i)-1} (1 - v_{ij}) = \sum_{i=0}^{N_r-1} f_s(i) N_l(i) \qquad (1)$$

where $N_l(i)$ is the number of VddL switches that can be achieved in $\mathcal{R}_i$. We assume that unused switches have been power-gated in this paper and therefore leakage power is not affected by Vdd-level assignment.

2.5 Problem Formulation

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

$$v_{ij} = v_{ik} \qquad 0 \le i < N_r \wedge 0 \le j, k < N_s(i) \qquad (2)$$

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

$$v_{ik} \le v_{ij} \qquad 0 \le i < N_r \wedge k \in \mathcal{FO}_{ij} \qquad (3)$$

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

$$a(v) \le T_{spec} \qquad \forall v \in \mathcal{PO} \qquad (4)$$

$$a(v) = 0 \qquad \forall v \in \mathcal{PI} \qquad (5)$$

$$a(u) + d(u, v) \le a(v) \qquad \forall u \in \mathcal{V} \wedge v \in \mathcal{FO}_u \qquad (6)$$

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 (7) is to maximize the power reduction (1).

$$Maximize \sum_{i=0}^{N_r-1} f_s(i) N_l(i) \qquad (7)$$

The tree based assignment problem consists of objective function(7), Vdd-level constraints (2) and timing constraints (4), (5) and (6). The segment based assignment problem is same as the tree based problem except that Vdd-level constraints (3) replace (2).

3. CHIP-LEVEL VDD ASSIGNMENT

3.1 Sensitivity Based Algorithms

Optimal Vdd-level assignment to circuit elements in a circuit is known to be NP-complete [11]. Below, we present two simple yet practical power sensitivity based heuristic algorithms, namely, *tree based heuristic* and *segment based heuristic*.

3.1.1 Tree Based Heuristic

Starting with a placed and routed single-Vdd circuit netlist, we calculate power sensitivity $\Delta P/\Delta V_{dd}$, which is the power reduction by changing VddH to 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 is performed to assign Vdd-level for routing trees. 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.

3.1.2 Segment Based Heuristic

The segment based heuristic is quite similar to the tree based heuristic except two differences. First, the assignment unit in the segment based heuristic 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 all of its fanout switches have been marked as 'tried' and assigned to VddL. 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. There is no performance loss in the segment based heuristic summarized in Figure 2.

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

Figure 2: Segment based heuristic.

3.2 Linear Programming Based Algorithm

The above algorithms implicitly allocate time slack first to routing trees or switches with higher transition density to reduce more power. Below, we present a *linear programming (LP) based algorithm* with explicit time slack allocation considering both global and local optimality. As the segment based assignment in general reduces more power than the tree based assignment, we only consider segment based assignment in the LP based algorithm that 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.

3.2.1 Chip-level Time Slack Allocation

A. 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 multiple of Δd, where Δd is the delay increase for an interconnect segment by changing the Vdd-level from VddH to VddL. Figure 3 presents a 3-pin routing tree as an example. $S0$ and $S1$ are the slacks allocated to two sinks $Sink0$ and $Sink1$, respectively. In Figure 3 (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 3 (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 3 (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 that is useful and slack more than the upper bound cannot lead to more VddL switches. For the rest part of the paper, we use slack to represent the useful slack.

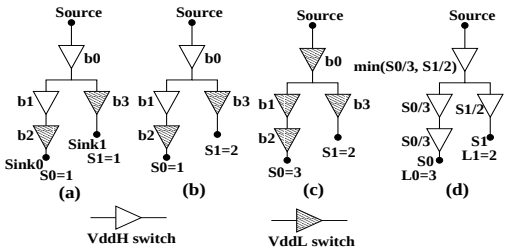


Figure 3: An example for estimating number of VddL switches.

Figure 3 (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

$$F_n(i) = \sum_{j=0}^{N_s(i)-1} min(\frac{s_{ik}}{l_{ik}} : \forall k \in \mathcal{SL}_{ij}) \quad (8)$$

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 3 (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 multiple of Δd, (8) gives a lower bound of number of VddL interconnect switches that can be achieved.*

Sketch of proof: It is easy to see that (8) gives the exact number of VddL switches for 2-pin tree. For a 3-pin tree, we can verify that (8) 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 $n-1$ pins, we can prove that it is true for any n-pin routing tree. □

B. LP Problem Formulation

The objective function (7) is to maximize power reduction which is the weighted sum of VddL switch number within each routing tree, where the weight is the transition density of each routing tree. To incorporate (8), 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

$$Maximize \sum_{i=0}^{N_r-1} f_s(i) F_n(i) \quad (9)$$

s.t.

$$F_n(i) = \sum_{j=0}^{N_s(i)-1} f_n(i,j) \qquad 0 \leq i < N_r \quad (10)$$

$$f_n(i,j) \leq \frac{s_{ik}-1}{l_{ik}} \quad 0 \leq i < N_r \wedge \forall k \in \mathcal{SL}_{ij} \quad (11)$$

The slack s_{ik} is a continuous variable normalized to Δd in (11) and also the rest part of the paper. To make (10) a lower bound of number of VddL switches, we replace $\frac{s_{ik}}{l_{ik}}$ with $\frac{s_{ik}-1}{l_{ik}}$ in (11) to avoid floor function $\lfloor s_{ik} \rfloor$ that is not a linear operation. The slack upper bound constraints can be expressed as

$$0 \leq s_{ik} \leq l_{ik} \qquad 0 \leq i < N_r \wedge 1 \leq k \leq N_k(i) \quad (12)$$

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

We modify the timing constraints (6) as follows. For the edges corresponding to routing in $\mathcal{G}$, the constraints considering slack can be expressed as

$$a(p_{i0}) + d(p_{i0}, p_{ik}) + s_{ik} \cdot \Delta d \leq a(p_{ik})$$
$$0 \leq i < N_r \wedge \forall p_{ik} \in \mathcal{FO}_{p_{i0}} \quad (13)$$

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

$$a(u) + d(u,v) \leq a(v) \qquad \forall u \in \mathcal{V} \wedge u \notin \mathcal{SRC} \wedge v \in \mathcal{FO}_u \quad (14)$$

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 (9), additional constraints (10) and (11), slack upper bound constraints (12), plus timing constraints (4), (5), (13) and (14). It is easy to verify that (4), (5) and (10) $\sim$ (14) are linear, and the objective function (9) 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 and the academia. In this paper, we use the LP solver from [12]. For the rest part of the paper, we use LP problem to represent the time slack allocation problem.

3.2.2 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. 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 slack left or no candidate switch in $\mathcal{R}_i$.

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

Sketch of proof: If 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 the same transition density, 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. □

3.2.3 Refinement

After net-level assignment, we may further reduce power by leveraging surplus slack. Figure 3(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 segment based heuristic (see Figure 2) to achieve more VddL switches and further reduce power.

4. EXPERIMENTAL RESULTS

In this section, we conduct experiments on the MCNC benchmark set and present the interconnect power and area reduction by the tree based heuristic, segment based heuristic and LP based algorithm compared to the baseline using Vdd-programmable interconnects with fine-grained Vdd-level converter inserted in routing channels[8]. We use the same Vdd-programmable interconnects in [8], but no level converter is inserted in routing channels. The unused interconnect switches are power-gated in either case. Same as [8], we use 1.3v for VddH and 0.8v for VddL in our experiments. We use the FPGA evaluation package *fpgaEva-LP* [2] to verify our power reduction. Because the power model in fpgaEva-LP is more accurate than the power model in our problem formulations, using fpgaEva-LP verifies both our modeling and problem formulations.

We present the interconnect power reduction achieved by the three algorithms in Table 2. Column 6 and Column 7 are the interconnect dynamic and leakage power for the baseline, respectively. By removing level converters in routing channels, we reduce interconnect leakage power by 91.78% (see column 8). The interconnect leakage power reduction is same for the three algorithms as this part of power reduction does not depend on Vdd-level assignment for interconnects. We can also reduce area by removing level converters and the associated configuration SRAM cells. As shown in Table 3, the interconnect device area is reduced by 25.48% compared to [8], where the area is represented in number of minimum width transistors.

circuit	w/ LCs baseline [8]	w/o LCs	area reduction
alu4	8027562	6031490	24.87%
apex2	12832956	9533624	25.71%
apex4	8807502	6559485	25.52%
bigkey	19520485	15065392	22.82%
clma	123197209	89125972	27.66%
des	28285474	21783998	22.99%
diffeq	8479705	6397439	24.56%
dsip	23769620	18101740	23.85%
elliptic	27411520	20210164	26.27%
ex1010	36205361	26561010	26.64%
ex5p	9176510	6825863	25.62%
frisc	57239492	41537145	27.43%
misex3	8587536	6430858	25.11%
pdc	53364989	38756961	27.37%
s298	10362364	7824915	24.49%
s38417	46594875	34463763	26.04%
s38584	37840516	28014506	25.97%
seq	12832956	9533624	25.71%
spla	30784702	22541947	26.78%
tseng	5911100	4484115	24.14%
avg.	-	-	25.48%

Table 3: Interconnect device area reduction by removing level converters in routing channels. Area is in number of minimum width transistors.

Column 2-5 in Table 2 present the percentage of VddL switches achieved by the three algorithms compared to the sensitivity based heuristic that uses a switch as an assignment unit in [8]. The tree based heuristic, segment based heuristic and LP based algorithm achieve 67.89%, 85.72% and 85.93% VddL switches, respectively. The segment based heuristic and LP based algorithm achieve almost the same VddL switches. Both of these two algorithms achieve more VddL switches than the tree based heuristic. In contrast, the sensitivity based heuristic in [8] achieved 83.97% VddL switches for Vdd-programmable interconnects with level converters. Both segment based heuristic and LP based algorithm achieve more VddL switches than [8] because we remove the delay overhead of level converters in the routing[2]. Column 9-11 presents the interconnect dynamic power achieved by the three algorithms. Compared to [8], the segment based heuristic and LP based algorithm reduce interconnect dynamic power by 1.92% and 4.68%, respectively. The tree based heuristic cannot reduce interconnect dynamic power compared to [8] as the assignment unit is a routing tree.

Column 12-14 in Table 2 present the overall interconnect power achieved by the three algorithms. Compared to [8], The tree based heuristic, segment based heuristic and LP based algorithm reduce interconnect power by 58.03%, 64.19% and 65.13%, respectively. The LP based algorithm achieves the best power reduction as it considers both global and local optimality. The segment based heuristic achieves slightly smaller power reduction compared to the LP based algorithm. Note that the timing specification may be relaxed for certain applications that are not timing-critical. In this case, our algorithms can achieve more VddL switches and therefore reduce more power with relaxed timing specification. The

[2]Without considering the delay overhead of level converters, [8] may achieve more VddL switches as the Vdd-programmable interconnects with level converters are more flexible in Vdd-level assignment.

1	2	3	4	5	6	7	8	9	10	11	12	13	14
	% of VddL switches				interconnect power		interconnect power w/o LCs compared to [8]						
		w/o LCs			w/ LCs (baseline)[8]		leakage	dynamic power			overall power		
circuit	w/ LCs [8]	tree based heuristic	segment based heuristic	LP based alg.	dynamic power (watt)	leakage power (watt)	power (due to LC removal)	tree based heuristic	segment based heuristic	LP based alg.	tree based heuristic	segment based heuristic	LP based alg.
alu4	69.12%	49.01%	73.70%	74.93%	0.03735	0.03444	-91.22%	+20.90%	-1.35%	-5.00%	-32.88%	-44.46%	-46.36%
apex2	76.97%	51.59%	80.50%	80.65%	0.04472	0.05665	-91.03%	+37.92%	+1.02%	-4.90%	-34.15%	-50.43%	-53.04%
apex4	71.44%	48.12%	73.24%	74.09%	0.02192	0.03863	-91.14%	+30.72%	+4.09%	-3.52%	-47.03%	-56.67%	-59.42%
bigkey	85.72%	80.70%	87.04%	87.78%	0.07125	0.08036	-92.92%	+3.62%	-1.33%	-1.42%	-47.55%	-49.87%	-49.92%
des	87.56%	76.81%	89.36%	89.61%	0.08156	0.11627	-93.83%	+8.40%	-2.18%	-2.96%	-51.68%	-56.04%	-56.37%
diffeq	93.89%	84.61%	94.01%	94.07%	0.00476	0.03666	-89.68%	-4.37%	-6.60%	-6.54%	-79.88%	-80.14%	-80.13%
dsip	86.56%	80.76%	87.55%	87.73%	0.07656	0.09994	-94.27%	+6.04%	-1.19%	-1.53%	-50.76%	-53.89%	-54.04%
elliptic	96.70%	88.34%	97.14%	97.09%	0.01716	0.12369	-91.81%	-4.96%	-6.16%	-8.38%	-81.23%	-81.38%	-81.65%
ex1010	77.61%	56.49%	81.01%	80.30%	0.03800	0.16439	-91.93%	+35.78%	-1.11%	-7.57%	-67.95%	-74.88%	-76.09%
ex5p	73.26%	53.16%	76.67%	75.44%	0.01968	0.04033	-91.54%	+16.45%	-2.17%	-3.75%	-56.13%	-62.23%	-62.75%
frisc	99.44%	97.17%	99.42%	99.45%	0.01251	0.26407	-93.74%	-11.93%	-11.33%	-12.18%	-90.03%	-90.01%	-90.04%
misex3	73.77%	47.95%	75.41%	75.94%	0.03653	0.03721	-91.07%	+27.48%	+2.20%	-2.83%	-32.34%	-44.86%	-47.35%
pdc	80.06%	53.66%	82.08%	82.22%	0.05591	0.24593	-92.92%	+36.01%	-3.21%	-5.74%	-69.04%	-76.31%	-76.78%
s298	87.42%	50.84%	88.67%	88.99%	0.01269	0.04383	-90.93%	+41.37%	-5.20%	-6.29%	-61.23%	-71.68%	-71.92%
s38417	90.76%	83.72%	92.04%	92.41%	0.06916	0.21047	-91.27%	+10.91%	+0.24%	-3.19%	-66.00%	-68.63%	-69.48%
s38584	98.07%	94.60%	98.39%	98.36%	0.06632	0.17088	-91.17%	+4.30%	-1.94%	-2.02%	-64.48%	-66.22%	-66.24%
seq	71.87%	48.12%	74.38%	75.32%	0.04767	0.05663	-91.59%	+28.07%	+1.06%	-5.03%	-36.90%	-49.24%	-52.03%
spla	77.64%	49.11%	80.28%	80.46%	0.04260	0.13954	-92.37%	+39.25%	+0.29%	-5.74%	-61.59%	-70.70%	-72.11%
tseng	97.63%	95.22%	97.75%	97.88%	0.00627	0.02527	-89.48%	-0.06%	-1.60%	-0.37%	-71.69%	-72.00%	-71.75%
avg.	83.97%	67.89%	85.72%	85.93%	-	-	-91.78%	+17.15%	-1.92%	-4.68%	-58.03%	-64.19%	-65.13%

Table 2: Percentage of VddL switches and interconnect power achieved by the three algorithms compared to the baseline using Vdd-programmable interconnects with level converters (LCs).

LP based algorithm achieves 96.21% VddL switches and reduces interconnect power by 72.52% when we relax critical path delay by 10% (see Table 4). Eventually VddL can be assigned to all switches and interconnect power reduction saturates if we allow sufficient critical path increase.

T_{spec} relaxation (%)	5%	10%	15%	20%	25%
% of VddL swithes	92.35%	96.21%	98.54%	99.79%	99.99%
interconnect power reduction (%) (baseline) [8]	69.75%	72.52%	74.00%	74.87%	75.21%

Table 4: Average percentage of VddL switches and power reduction achieved by the LP based algorithm with relaxed time specification.

circuit	# of nodes	runtime (s)		
		tree based	segment based	LP based
alu4	10716	60.52	124.4	482.53
apex2	14860	180.75	378.59	1153.28
apex4	9131	66.93	177.52	461.37
clma	91620	8763.24	16799.67	>20H
elliptic	30192	607.85	913.04	3136.59
ex1010	33265	836.32	1422.79	5109.22
frisc	40662	1135.84	1912.15	6135.38
pdc	40001	1254.57	2508.57	8210.07
s38417	57503	1821.09	2895.79	9152.52
s38584	46014	1255.31	1892.86	6863.62
geometric mean		$1X$	$1.85X$	$6.66X$

Table 5: Runtime comparison between the three algorithms.

Table 5 compares the runtime [3] between the three algorithms. The tree based heuristic is the fastest among the three algorithms. The segment based heuristic and the LP based algorithm take $1.85X$ and $6.66X$ runtime compared to the fastest one. For the largest circuit *clma*, the LP based algorithm cannot solve the LP problem after running 20 hours. Compared to the LP based algorithm, the segment based algorithm has slightly smaller power reduction, but runs $4X$ faster and is effective for large circuits. The LP based algorithm is worthwhile for small circuits and can achieve best power reduction.

5. CONCLUSIONS

We have developed chip-level dual-Vdd assignment algorithms to guarantee that no VddL switch drives VddH switches. This removes the need of Vdd-level converters and reduces interconnect leakage and interconnect device area by 91.78% and 25.48% respectively compared to [8]. We have presented two simple yet practical power sensitivity based heuristics, *tree based heuristic* and *segment based heuristic*, which implicitly allocate time slack first to interconnects with higher transition density and assign VddL to them for more power reduction. We have also presented a *linear programming (LP) based algorithm* in which time slack is first explicitly allocated to each routing tree by formulating the problem as an LP problem to maximize a lower bound of power reduction, and then the Vdd-level assignment is solved optimally within each routing tree given the allocated time slack. We have conducted the experiments on MCNC benchmark set and compared the power reduction by the three algorithms. Compared to [8], the LP based algorithm obtains the best power reduction and reduces interconnect power by 65.13% without performance loss. The tree based heuristic and segment based heuristic reduce interconnect power by 58.03% and 64.19%, respectively. Compared to the LP based algorithm, the segment based heuristic has slightly smaller power reduction, but runs $4X$ faster and is effective for large circuits. In the future, we will study power-driven routing, which simultaneously performs routing and Vdd-level assignment for Vdd-programmable interconnects.

[3] The runtime includes single-Vdd placement and routing by VPR and generating the interface file between VPR and fpgaEva-LP.

Acknowledgement

The authors like to thank Mr. Fei Li and Mr. Jinjun Xiong for helpful discussions.

6. REFERENCES

[1] K. Poon, A. Yan, and S. Wilton, "A flexible power model for FPGAs," in *Proc. of Intl. conf. on Field-Programmable Logic and Applications*, Sept 2002.

[2] F. Li et al, "Architecture evaluation for power-efficient FPGAs," in *FPGA Symp.*, Feb 2003.

[3] J. H. Anderson et al, "Active leakage power optimization for FPGAs," in *FPGA Symp.*, Feb 2004.

[4] J. Lamoureux and S. J. Wilton, "On the interaction between power-aware FPGA CAD algorithms," in *ICCAD*, Nov 2003.

[5] A. Gayasen et al, "Reducing leakage energy in FPGAs using region-constrained placement," in *FPGA Symp.*, Feb 2004.

[6] F. Li et al, "Low-power FPGA using pre-defined dual-vdd/dual-vt fabrics," in *FPGA Symp.*, Feb 2004.

[7] F. Li, Y. Lin, and L. He, "FPGA power reduction using configurable dual-vdd," in *DAC*, June 2004.

[8] F. Li, Y. Lin, and L. He, "Vdd programmability to reduce FPGA interconnect power," in *ICCAD*, Nov 2004.

[9] J. H.Anderson and F. N.Najm, "Low-power programmable routing circuitry for FPGAs," in *ICCAD*, Nov 2004.

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

[11] A. Gayasen et al, "A dual-vdd low power FPGA architecture," in *Proc. Intl. Conf. Field-Programmable Logic and its Application*, August 2004.

[12] M Berkelaar, *lp-solver 3.2: a public domain (MI)LP solver*. ftp://ftp.ics.ele.tue.nl/pub/lp_solve/.