An Efficient Chip-level Time Slack Allocation Algorithm for Dual-Vdd FPGA Power Reduction *

Yan Lin[1], Yu Hu[1], Lei He[1] and Vijay Raghunat[2]
Electrical Engineering Dept., UCLA, Los Angeles, CA[1]
Purdue University, La Fayette, IN[2]

ABSTRACT

To reduce FPGA power, a linear programming (LP) based time slack allocation algorithm, EdTLC-LP, has been proposed recently for Vdd-programmable interconnects without using Vdd-level converters for mixed wire lengths. However, it takes a long time to solve the LP problem for time slack allocation. In this paper, we develop EdTLC-NW, a slack allocation algorithm based on min-cost network flow to reduce runtime. Compared to single Vdd FPGA with power-gating, EdTLC-LP and EdTLC-NW reduce interconnect power by 52.71% and 52.52%, respectively. EdTLC-NW achieves as good results as EdTLC-LP but runs $8X$ faster on average. Furthermore, the speedup increases for larger circuits and EdTLC-NW is $20X$ faster for the largest circuit.

Categories and Subject Descriptors: B.7.2 [Integrated Circuits]: Design aids
General Terms: Algorithms, Design
Keywords: Low power, time slack, FPGA

1. INTRODUCTION

FPGA power modeling and reduction has become an active research area recently. [1, 2] present power evaluation frameworks for generic parameterized FPGA architectures, and show that both interconnect and leakage power are significant for nanometer FPGAs. [3] studies the interaction of a suite of power-aware FPGA CAD algorithms without changing the existing FPGAs. [4] proposes a configuration inversion method to reduce leakage power of multiplexers. In addition, dual-Vdd and Vdd programmability have been applied to FPGA to reduce power. [5, 6] are the first work introducing dual-Vdd and field programmability of Vdd to FPGA. Vdd programmability has been applied to both FPGA logic blocks [5, 6] and interconnects [7, 8, 9].

A Vdd-level converter is needed when a low-Vdd (VddL) circuit elements drives a high-Vdd (VddH) circuit element to avoid excessive leakage. [8] inserts a level converter in front of each interconnect switch to provide the fine-grained Vdd programmability for interconnects. However, it has been shown in [10] that this fine-grained Vdd-level converter insertion may introduce large leakage and area overhead. Recently, a few approaches have been presented without directly using level converters in Vdd-programmable interconnects. [9] uses the positive feedback PMOS transistor in the level-restore buffer as an alternative level converter with much reduced area and power overhead. [7] enforces that all the routing trees driven by (driving) a logic block have the same Vdd-level as the source (sink) logic block when level converters are inserted at CLB inputs (outputs). [11] uses a smaller granularity, a routing tree, as the unit in Vdd-level assignment. [10] further allows a mix of Vdd-levels within a routing tree, but only VddH switches can drive VddL switches. [12] extends the algorithms in [10] for mixed interconnect wire lengths.

In this paper, we use the same circuit design from [10, 12] and aim to improve the linear programming (LP) based time slack allocation algorithm, *EdTLC-LP*. In EdTLC-LP, time slack is allocated to each routing tree by formulating the problem as an LP problem to minimize power. However, it takes unacceptable runtime for EdTLC-LP to solve the LP problem for time slack allocation (more than 10 hours for the largest circuit *clma* on a 1.9GHz Xeon machine). Our contribution is to formulate the time slack allocation problem as a min-cost network flow problem and present a new algorithm, *EdTLC-NW* , which significantly reduces the run-time. Using single-Vdd FPGA with power-gating as the baseline, EdTLC-LP and EdTLC-NW reduce interconnect power by 52.71% and 52.52%, respectively. EdTLC-NW achieves as good results as EdTLC-LP but runs $8X$ faster on average. Furthermore, the speedup increases for larger circuits and EdTLC-NW achieves up to $20X$ speedup in overall runtime.

The rest of the paper is organized as follows. Section 2 introduces background and modeling. Section 3 reviews the LP based budgeting. Section 4 describes netflow based budgeting for interconnects with mixed wire lengths. Section 5 discusses the experimental results. Section 6 concludes this paper.

2. PRELIMINARIES

2.1 Delay and Power Modeling with Dual-Vdd

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.

A directed acyclic timing graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ [13] is constructed to model the circuit for timing analysis. The Elmore delay model is used to calculate the routing delay. We define the

*This paper is partially supported by NSF grant CCR-0306682. Address comments to lhe@ee.ucla.edu.

ISLPED'06, October 4–6, 2006, Tegernsee, Germany.

$\mathcal{G}(\mathcal{V},\mathcal{E})$	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
c_{ij}	load capacitance 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 $\mathcal{R}_i$
$f_s(i,j)$	transition density of j^{th} switch in $\mathcal{R}_i$
$N_k(i)$	# 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 $\mathcal{R}_i$
W_{ik}	power weight associated with k^{th} sink in $\mathcal{R}_i$
w_{ik}	power weight associated with $e(i,j)$ in $\mathcal{G}$

Table 1: Notations frequently used in this paper

fanout cone of a switch as the sub-tree of the routing tree rooted at the switch. Dynamic power occurs 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. Let v_{ij} indicate 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

$$P_r = \sum_{i=0}^{N_r-1} \sum_{j=0}^{N_s(i)-1} (1-v_{ij})(0.5 f_{clk} f_s(i,j) c_{ij} \Delta Vdd^2 + \Delta P_s(i,j)) \quad (1)$$

which is the sum of dynamic and leakage power reduction. N_r is the total number of routing trees, $f_s(i,j)$ is the transition density of j^{th} switch in i^{th} routing tree $\mathcal{R}_i$, $N_s(i)$ is the number of switches in $\mathcal{R}_i$, and $\Delta P_s(i,j)$ and c_{ij} are the leakage power reduction and load capacitance of each switch, respectively.

Dual-Vdd tree based level converter insertion [10, 12] is used in this paper. A mix of Vdd-levels within one routing tree is allowed, and the Vdd-level constraints are

$$v_{ik} \le v_{ij} \qquad 0 \le i < N_r \wedge 0 \le j < N_s(i) \wedge k \in \mathcal{FO}_{ij} \quad (2)$$

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

3. LINEAR PROGRAMMING BASED BUDGETING

In this section, we review the LP based time slack allocation algorithm, *EdTLC-LP*, for mixed interconnect wire lengths [12]. Time slack is first allocated to each routing tree by formulating the problem as an LP problem considering the load capacitance of each switch explicitly. A bottom-up assignment algorithm is then performed to achieve the optimal solution within each routing tree for the allocated time slack. A refinement step is finally performed to leverage surplus time slack.

3.1 Estimation of Interconnect Power Reduction

Estimating power reduction given the allocated slack is the key for the LP and netflow based algorithms. There is an upper bound for slack, which is the delay increase when VddL is assigned to all the switches in a tree. Clearly, slack more than the upper bound cannot lead to more VddL switches. The slack upper bound constraints can be expressed as

$$0 \le S_{ik} \le D_{ik} \qquad 0 \le i < N_r \wedge 1 \le k \le N_k(i) \quad (3)$$

where $N_k(i)$ is the number of sinks in $\mathcal{R}_i$ and D_{ik} is the delay increase of the path from the source to k^{th} sink in $\mathcal{R}_i$ when VddL is assigned to all the switches in that path.

Let l_{ik} represent the number of switches in the path from the source to the k^{th} sink in $\mathcal{R}_i$. Slack S_{ik} is first transformed into s_{ik}, which is expressed in number of switches as follows,

$$s_{ik} = \frac{S_{ik}}{D_{ik}} \cdot l_{ik} \quad (4)$$

Let c_{ij} represent the load capacitance of the j^{th} switch in $\mathcal{R}_i$. Let C_{ik} represent the total load capacitance of the switches in the path from the source to the k^{th} sink in $\mathcal{R}_i$. *Sink list* $\mathcal{SL}_{ij}$ is defined as the set of sinks in the fanout cone of the j^{th} switch in $\mathcal{R}_i$. The number of VddL switches given the allocated slack is then estimated as

$$F_n(i) = \sum_{j=0}^{N_s(i)-1} min(\frac{s_{ik}}{C_{ik}} \cdot c_{ij} : \forall k \in \mathcal{SL}_{ij}) \quad (5)$$

The rationale is that we consider k^{th} sink with minimum $s_{ik}c_{ij}/C_{ik}$ in sink list $\mathcal{SL}_{ij}$ as the most critical sink to j^{th} switch in $\mathcal{R}_i$.

The dynamic/leakage power reduction of the tree $\mathcal{R}_i$ is estimated as the sum of the dynamic/leakage power reduction of each switch in $\mathcal{R}_i$ and can be expressed as,

$$P_{dr}(i) = 0.5 f_{clk} \cdot \Delta Vdd^2 \sum_{j=0}^{N_s(i)-1} [min(\frac{s_{ik}}{C_{ik}} \cdot c_{ij} : \forall k \in \mathcal{SL}_{ij}) \cdot f_s(i,j) \cdot c_{ij}] \quad (6)$$

$$P_{lr}(i) = \sum_{j=0}^{N_s(i)-1} [min(\frac{s_{ik}}{C_{ik}} \cdot c_{ij} : \forall k \in \mathcal{SL}_{ij}) \cdot \Delta P_s(i,j)] \quad (7)$$

where $\Delta P_s(i,j)$ is the leakage power difference of j^{th} switch in $\mathcal{R}_i$ between VddH and VddL. Wire segments with different lengths might be driven by switches with different sizes.

3.2 LP Problem Formulation

Similar to [10], the net-based formulation is used, 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

$$a(v) \le T_{spec} \qquad \forall v \in \mathcal{PO} \quad (8)$$

$$a(v) = 0 \qquad \forall v \in \mathcal{PI} \quad (9)$$

$$a(u) + d(u,v) \le a(v) \qquad \forall u \in \mathcal{V} \wedge v \in \mathcal{FO}_u \quad (10)$$

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

The objective is to maximize interconnect power reduction given by the sum of (6) and (7). To incorporate them 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

$$Maximize \sum_{i=0}^{N_r-1} 0.5 f_{clk} \Delta V dd^2 \sum_{j=0}^{N_s(i)-1} f_n(i,j) f_s(i,j) c_{ij} + \sum_{i=0}^{N_r-1} \sum_{j=0}^{N_s(i)-1} f_n(i,j) \Delta P_s(i,j) \quad (11)$$

s.t.

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

The timing constraints (10) is then modified as follows. For the edges corresponding to routing in $\mathcal{G}$, the constraints considering slack can be expressed as

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

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 $\mathcal{G}$, S_{ik} is the slack allocated to k^{th} sink in $\mathcal{R}_i$ 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 $\mathcal{SRC}$ contains vertices corresponding to routing tree sources.

The *time slack allocation problem* is formulated using objective function (11), additional constraints (12), slack upper bound constraints (3), and timing constraints (8), (9), (13) and (14). It is easy to verify that all the constraints are linear, and the objective function (11) is also linear.

THEOREM 1. *The time slack allocation problem is a linear programming (LP) problem.*

4. NETWORK FLOW BASED BUDGETING

The runtime of time slack allocation in EdTLC-LP can be very long for large circuits mainly due to the expensive computational time of linear programming. In this section, we formulate the time slack allocation problem as a min-cost network flow problem and present a new algorithm, *EdTLC-NW* , with significantly reduced runtime.Similar network flow formulation has been used for timing budgeting in high level synthesis [14].

4.1 Network Flow Formulation

We first deliberately distribute slack s_{ik} to the switches in the path from the source to to k^{th} sink in $\mathcal{R}_i$. As the min function in (5) cannot be efficiently handled by a network flow formulation, we define the sink with the minimum slack as the *critical sink* in the fanout cone of a switch with multiple sinks. We then use $s_{ik} c_{ij} / C_{ik}$ of this critical sink to replace the min operator over all sinks in the fanout cone of a switch. The dynamic power reduction (6) and leakage power reduction (7) for $\mathcal{R}_i$ can be rewritten as follows,

$$P_{dr}(i) = 0.5 f_{clk} \cdot \Delta V dd^2 \cdot \sum_{j=0}^{N_s(i)-1} S_{ik} \cdot [f_s(i,j) \cdot \frac{l_{ik} \cdot c_{ij}^2}{D_{ik} \cdot C_{ik}}] \quad (15)$$

$$P_{lr}(i) = \sum_{j=0}^{N_s(i)-1} S_{ik} \cdot [(\frac{l_{ik} \cdot c_{ij}}{D_{ik} \cdot C_{ik}}) \cdot \Delta P_s(i,j)] \quad (16)$$

The objective function can be rewritten as following by merging the coefficient of the slack S_{ik} of k^{th} sink in $\mathcal{R}_i$ as follows,

$$Maximize \quad \sum_{i=0}^{N_r-1} \sum_{k=0}^{N_k(i)-1} W_{ik} \cdot S_{ik} = \sum_{\forall Sink} W_{ik} \cdot S_{ik} \quad (17)$$

$$W_{ik} = \sum_{\forall j \in \mathcal{UBC}_{ik}} [0.5 f_{clk} \Delta V dd^2 c_{ij} f_s(i,j) + \Delta P_s(i,j)] \cdot \frac{c_{ij} \cdot D_{ik}}{(C_{ik} \cdot l_{ik})} \quad (18)$$

where set $\mathcal{UBC}_{ik}$ include all switches with k^{th} sink as the critical sink in $\mathcal{R}_i$.

Since $W_{ik} > 0$ for all sinks, we can restrict timing constraint (13) as the following equation to maximize the objective function,

$$S_{ik} = a(p_{ik}) - a(p_{i0}) - d(p_{i0}, p_{ik}), \qquad 0 \leq i < N_r \wedge \forall p_{ik} \in \mathcal{FO}_{p_{i0}} \quad (19)$$

After substituting S_{ik} using (19) and rearrangement, objective function (17) can be expressed as,

$$Maximize \quad \sum_{i=0}^{N_r-1} \sum_{k=0}^{N_k(i)-1} W_{ik} \cdot [a(p_{ik}) - a(p_{i0}) - d(p_{i0}, p_{ik})] \quad (20)$$

Similarly, slack bound constraint (3) can be rewritten as

$$a(p_{i0}) - a(p_{ik}) \leq -d(p_{i0}, p_{ik}) \quad (21)$$

$$a(p_{ik}) - a(p_{i0}) \leq d(p_{i0}, p_{ik}) + D_{ik} \quad (22)$$

We then merge the timing constraint (21) and (14) into the general expression (10).

Similar to [14], a virtual input node (SI) and a virtual output node (SO) are added into $\mathcal{G}$ to connect all nodes in $\mathcal{PI}$ and $\mathcal{PO}$, respectively. All edges connected to SI and SO have zero delay. We add a backward edge $e(p_{ik}, p_{i0})$ for each source sink pair in $\mathcal{R}_i$. A delay of $-(d(p_{i0}, p_{ik}) + D_{ik})$ is associated to $e(p_{ik}, p_{i0})$ to represent the slack upper bound. A virtual edge $e(SO, SI)$ with delay $-T_{spec}$ is then added. All constraints can now be represented by edges in $\mathcal{G}$. For example, edge $e(u,v)$ with delay $d(u,v)$ represents constraint $a(u) - a(v) < -d(u,v)$.

To represent the objective function (20) in $\mathcal{G}$, we associate a weight w_{uv} in each edge $e(u,v)$. For those edges $e(p_{i0}, p_{ik})$ corresponding to routing, let $w_{p_{i0}p_{ik}} = W_{ik}$. For other edges, let $w_{uv} = 0$. The objective function (20) can then be rewritten as,

$$Maximize \quad \sum_{v \in V} a(v) (\sum_{u \in \mathcal{FI}_v} w_{uv} - \sum_{u \in \mathcal{FO}_v} w_{vu}) - \sum_{i=0}^{N_r-1} \sum_{k=0}^{N_k(i)-1} d(p_{i0}, p_{ik}) \quad (23)$$

where $\sum_{i=0}^{N_r-1} \sum_{k=0}^{N_k(i)-1} d(p_{i0}, p_{ik})$ is a constant and can be removed from the objective function (23), and $\mathcal{FI}_v / \mathcal{FO}_v$ is fanin/fanout set of vertex v.

For the optimization problem with constraints (10) and (22) and objective function (23), its dual problem is

$$\min \quad \sum_{e_(i,j) \in \mathcal{E}} (d(i,j) + D_{ij}) \cdot z_{ij} - d(i,j) \cdot y_{ij} \quad (24)$$

$$s.t. \quad \sum_{e(k,i) \in \mathcal{E}} (y_{ki} - z_{ki}) - \sum_{e(i,j) \in \mathcal{E}} (y_{ij} - z_{ij}) = \rho_i \quad (25)$$

$$\rho_i = \sum_{j \in \mathcal{FI}_i} w_{ji} - \sum_{k \in \mathcal{FO}_k} w_{ik} \quad (26)$$

$$y_{ij}, z_{ij} \in R_+ \quad (27)$$

To verify that the above dual problem is a min-cost network flow problem on $\mathcal{G}$, y_{ij} is the flow along $e(i,j)$ with cost $-d(i,j)$, z_{ij} is the flow along $e(j,i)$, which corresponds to routing and is associated with cost $d(i,j)+D_{ij}$. Obviously, no negative cycle is introduced by the backward edges. ρ_i is the demand in each vertex. Note that $\sum_{i\in\mathcal{V}}\rho_i = 0$ is satisfied as required in the min-cost network flow problem. Hence, we have the following theorem.

THEOREM 2. *The dual problem of the time slack allocation problem is a min-cost network flow problem.*

After solving the min-cost network flow problem, we can get the solutions for variables y_{ij} and z_{ij}. Similar to [14], we can calculate the solution of the primal problem. We first construct the residual graph $\mathcal{G}'(\mathcal{V},\mathcal{E}')$ from the original $\mathcal{G}$. For any edge $e(i,j)$ in $\mathcal{G}'$ with non-zero flow, there are two edges $e(j,i)$ and $e(i,j)$ in $\mathcal{G}'$. The cost of each backward edge $e(j,i)$ is $d(i,j)$, and is equal to the complement of the forward edge cost. Let δ_i be the shortest distance from SI to vertex i in $\mathcal{G}'$. It has been proved in [14] that $a(i) = -\delta_i$ is an optimal solution to the primal problem. We use the push-relabel algorithm [15] for min-cost flow problem and Bellman-Ford algorithm [16] for shortest path problem.

4.2 Comparison Between EdTLC-LP and EdTLC-NW

The basic difference between EdTLC-LP and EdTLC-NW is the way to calculate the slack for each switch. EdTLC-LP chooses the minimum $s_{ik}c_{ij}/C_{ik}$ among all sinks while EdTLC-NW chooses the $s_{ik}c_{ij}/C_{ik}$ of the critical sink. Let k^{th} sink be the critical sink in the fanout cone of j^{th} switch in $\mathcal{R}_i$. EdTLC-NW estimates number of VddL switches in $\mathcal{R}_i$ as

$$F_n(i) = \sum_{j=0}^{N_s(i)-1} \frac{s_{ik}}{C_{ik}} \cdot c_{ij} \tag{28}$$

It has been proved in [10] that (5) can always give a lower bound of the number of VddL switches that can be achieved in $\mathcal{R}_i$ for uniform wire length. However, (5) cannot always give an infimum (the greatest lower bound) for the VddL switch number. Figure 1 (a) shows a simple example (uniform length of wire segments is assumed for simplicity).

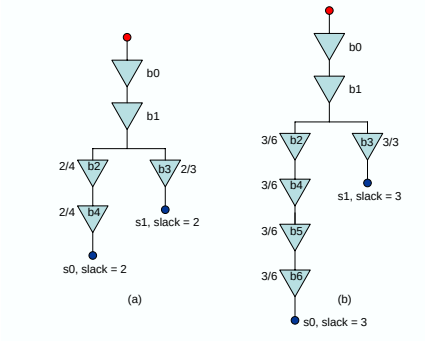

Figure 1: An example of estimated VddL switch #

Suppose $S1$ is the critical sink based on the current timing graph, and a slack of 2 is allocated to both $S0$ and $S1$. (5) estimates VddL switch number as $(2/4 + 2/4 + 2/3 + 2\min(2/4 + 2/3)) = 8/3$ while (28) gives an estimation as $(2/4 + 2/4 + 2/3 + 2 \cdot 2/3) = 3$. Obviously, we CAN achieve three VddL switches while satisfying the allocated slack $S0 = S1 = 2$. This indicates that EdTLC-LP cannot always give the infimum of VddL switch number.

On the other hand, EdTLC-NW always gives a greater estimation than EdTLC-LP while it might give an overestimated result. As shown in Figure 1 (b), suppose $S1$ is still the critical sink. The slacks allocated to $S0$ and $S1$ are both 3. (5) gives an estimation as $(3/6 + 3/6 + 3/6 + 3/6 + 3/3 + 2\min(3/6 + 3/3)) = 4$ while (28) gives $(3/6 + 3/6 + 3/6 + 3/6 + 3/3 + 2 \cdot 3/3) = 5$. However, we cannot achieve five VddL switches while satisfying the allocated slacks $S0 = S1 = 3$. This indicates that EdTLC-NW may overestimate the VddL switch number in some cases.

In summary, EdTLC-LP may give a conservative estimation of power savings while EdTLC-NW may give an over-optimistic estimation. In the experiments to be presented in Section 5, the two formulations give a similar estimation for most cases due to the fact that the most critical sink is usually the one with the minimum $s_{ik}c_{ij}/C_{ik}$. In addition, EdTLC-NW can achieve results close to EdTLC-LP in terms of power reduction but with significantly shorter runtime.

5. EXPERIMENTAL RESULTS

5.1 Experimental Settings

We conduct the experiments on the largest MCNC benchmarks [17] including ten combinational circuits (group I in Table 2) and ten sequential circuits (group II in Table 2). We map them into FPGA with LUT size of 4 and cluster size of 10. We use the same Vdd-programmable logic blocks and interconnects in [10], but with a mix of different interconnect wire lengths. We use 60% length 4 wire and 40% length 8 wire for better performance and area tradeoff, as suggested in [18]. The unused interconnect switches are power-gated in all cases. Similar to [8], we customize the FPGA chip size for each benchmark circuit and use the smallest chip that fits each benchmark. We use 1.3v for VddH and 0.8v for VddL same as [10] in our experiments at $100nm$ technology node. We perform dual-Vdd assignment without delay increase compared to the circuit using only VddH in the rest of the paper.

We first use VPR [13] for single-Vdd placement and routing. Before applying budgeting algorithms to the Vdd programmable interconnects, a sensitivity based assignment [6] is first performed to assign Vdd-level for Vdd-programmable logic blocks without performance loss[1]. The cycle-accurate FPGA power simulator *fpgaEva-LP2* [11] is then used to calculate power.

5.2 Comparison of Interconnect Power

We first compare the the number of VddL switches achieved by EdTLC-LP and EdTLC-NW in Table 2, as it is a good indication of power reduction. The number of VddL switches is expressed in percent of used switch number. EdTLC-LP and EdTLC-NW achieve 84.43% and 84.05% VddL switches, respectively. Both achieve almost the same number of VddL switches.

[1]Both algorithms EdTLC-LP and EdTLC-NW do consider the fact that VddL logic blocks consume time slack.

We then present the interconnect power reduction achieved by EdTLC-LP and EdTLC-NW in Table 3. Single-Vdd FPGA with power-gating is used as the baseline for interconnect power. Columns 2-3 present the interconnect dynamic and leakage power for the baseline case. Columns 4-5 present the overall interconnect power reduction for EdTLC-LP and EdTLC-NW . Compared to the baseline case, EdTLC-LP and EdTLC-NW reduce total interconnect power by 52.71% and 52.03%, respectively. We also present the interconnect dynamic power and leakage power reduction in columns 6-9. Compared to the baseline, EdTLC-LP and EdTLC-NW reduce dynamic power by 52.03% and 51.69%, and leakage power by 62.71% and 62.51%, respectivly. Clearly, EdTLC-NW achieves as good results as EdTLC-LP .

For both algorithms, we also present the contribution of refinement step in Table 2 and Table 3. The refinement step achieves 3.49% and 4.17% VddL switches for EdTLC-LP and EdTLC-NW , respectively. Compared to baseline, the refinement step in EdTLC-LP /EdTLC-NW obtains 3.35%/ 1.03% more power reduction, 3.45%/0.96% more dynamic power reduction and 2.23%/2.57% more leakage power reduction, respectively. It is clear that the refinement step is effective to distribute surplus time slack and further reduce interconnect power. The first source of the surplus time slack is the difference between the continuous problem formulations (i.e., the allocated slack is continuous in budgeting) and the fact that Vdd-level assignment is discrete (i.e., the slack consumed by a VddL switches must be Δd). Secondly, the objective of our formulations is to maximize the estimated power reduction instead of the exact power reduction. The two may be different due to the fact that not all allocated slack is useful for power reduction.

	Circuit	Cluster#	EdTLC-LP	EdTLC-NW
I	ex5p	123	68.32% (3.43%)	67.99% (4.87%)
	apex4	134	75.38% (6.65%)	73.27% (8.04%)
	misex3	153	73.77% (4.12%)	72.41% (9.11%)
	alu4	162	78.67% (4.89%)	78.08% (6.54%)
	seq	198	70.49% (4.93%)	69.52% (6.63%)
	apex2	213	78.48% (3.98%)	78.09% (4.74%)
	des	218	82.13% (1.94%)	82.01% (1.86%)
	spla	399	78.46% (4.40%)	78.36% (6.01%)
	ex1010	493	78.77% (5.55%)	78.59% (6.84%)
	pdc	568	79.76% (4.19%)	79.29% (5.16%)
II	tseng	131	97.02% (2.58%)	97.07% (0.71%)
	dsip	162	91.70% (1.10%)	91.67% (1.67%)
	diffeq	195	92.93% (2.58%)	92.90% (1.21%)
	s298	256	89.62% (2.93%)	89.59% (1.41%)
	bigkey	294	81.43% (2.18%)	80.88% (7.27%)
	elliptic	421	98.72% (1.92%)	98.75% (0.28%)
	frisc	595	99.19% (4.74%)	99.19% (2.58%)
	s38584.1	704	97.77% (1.86%)	97.75% (0.65%)
	s38417	847	90.15% (2.53%)	89.86% (2.93%)
	clma	1358	85.93% (3.20%)	85.66% (3.38%)
	Ave	266	84.43% (3.49%)	84.05% (4.17%)

Table 2: Percentage of VddL switches achieved by EdTLC-LP and EdTLC-NW

5.3 Comparison of Runtime

Table 4 compares the runtime of EdTLC-LP and EdTLC-NW. Column "budget" refers to the runtime for distributing slack to each tree. The simplex method based LP solver [19] is used in EdTLC-LP . Column "total" presents the overall runtime. Column "speedup" shows the ratio of speedup achieved by EdTLC-NW compared to EdTLC-LP. EdTLC-NW achieves as good results as EdTLC-LP but runs $8X$ faster on average. It is clear that the speedup of the budgeting time increases for the larger circuits. For the largest circuit *clma*, which contains over 1k clusters, EdTLC-NW achieves up to $6000X$ speedup in time slack allocation and $20X$ speedup in overall runtime. The current commercial FPGA designs often contain 10k to 100k clusters [20], which indicates that the EdTLC-NW may have more speedup than that reported in this paper in practice. The min-cost network flow based algorithm EdTLC-NW takes negligible runtime in time slack allocation compared to EdTLC-LP . Clearly, the efficiency of EdTLC-NW makes our algorithm highly scalable, especially for the applications that need iterative budgeting for different time specifications.

	EdTLC-NW		EdTLC-LP		speedup(X)	
cir	budget	total	budget	total	budget	total
ex5p	1	33	1187	1254	118x	5x
apex4	1	33	184	228	184x	7x
misex3	1	38	155	227	155x	6x
alu4	1	36	101	131	101x	4x
seq	1	55	179	216	179x	4x
apex2	1	78	413	502	413x	6x
des	1	95	327	444	327x	5x
spla	1	300	837	1181	837x	4x
ex1010	2	346	2391	2804	1196x	8x
pdc	2	803	3633	4496	1817x	6x
tseng	1	32	83	91	83x	3x
dsip	1	44	181	223	181x	5x
diffeq	1	41	252	321	252x	8x
s298	1	96	371	494	371x	5x
bigkey	1	89	478	589	478x	7x
elliptic	1	229	877	1128	877x	5x
frisc	2	479	1823	2364	912x	5x
s38584	3	421	2305	2806	768x	7x
s38417	4	709	3719	4463	930x	6x
clma	9	2735	53712	56726	5968x	21x
Ave	1	335	3660	2775	3607x	8x

Table 4: Runtime (second) comparison between EdTLC-LP and EdTLC-NW

6. CONCLUSIONS

To reduce power in dual-Vdd FPGA, we have re-formulated the LP based time slack allocation problem to a min-cost network flow based problem and presented a new network flow based algorithm, EdTLC-NW , with significantly shorter run-time. Using single-Vdd FPGA with power-gating as the baseline, the linear programming (LP) based time budgeting algorithm EdTLC-LP [12] and EdTLC-NW reduce interconnect power by 52.71% and 52.52%, respectively. EdTLC-NW achieves as good results as EdTLC-LP but runs $8X$ faster on average. The speedup increases for larger circuits. For the largest circuit, EdTLC-NW achieves up to $6000X$ speedup in time slack allocation and $20X$ speedup in overall runtime. Clearly, the efficiency of EdTLC-NW makes our algorithm highly scalable, especially for the applications that may need iterative budgeting procedures. We expect that EdTLC-NW has more speedup in real designs than that reported in this paper since real designs are often bigger than the examples in this paper.

7. 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.

1	2	3	4	5	6	7	8	9
	baseline power (W)		interconnect power reduction		dynamic power reduction		leakage power reduction	
	dynamic	leakage	EdTLC-LP	EdTLC-NW	EdTLC-LP	EdTLC-NW	EdTLC-LP	EdTLC-NW
ex5p	0.017687	0.000744	40.64% (3.25%)	40.42% (1.31%)	40.31% (3.28%)	40.08% (1.22%)	48.33% (2.55%)	48.47% (3.47%)
apex4	0.020157	0.000768	45.24% (4.92%)	44.19% (1.49%)	44.91% (4.93%)	43.87% (1.34%)	53.79% (4.81%)	52.38% (5.19%)
misex3	0.037557	0.000768	44.57% (3.23%)	44.02% (2.73%)	44.40% (3.23%)	43.86% (2.65%)	52.83% (2.89%)	52.14% (6.70%)
alu4	0.031968	0.000617	48.01% (6.09%)	47.92% (1.55%)	47.83% (6.14%)	47.73% (1.50%)	57.66% (3.34%)	57.45% (4.15%)
seq	0.053091	0.001042	45.43% (3.58%)	45.06% (1.65%)	45.31% (3.59%)	44.94% (1.60%)	51.71% (2.96%)	51.25% (3.95%)
apex2	0.053479	0.00121	49.71% (4.35%)	49.56% (0.95%)	49.51% (4.39%)	49.36% (0.90%)	58.65% (2.62%)	58.33% (2.97%)
des	0.069708	0.001174	50.18% (1.31%)	50.02% (0.74%)	50.02% (1.30%)	49.86% (0.73%)	59.73% (1.78%)	59.80% (1.78%)
spla	0.042551	0.002538	49.88% (3.74%)	49.75% (1.37%)	49.39% (3.80%)	49.27% (1.24%)	58.09% (2.65%)	57.83% (3.68%)
ex1010	0.028371	0.003003	46.08% (4.96%)	45.89% (1.38%)	44.74% (5.09%)	44.53% (1.03%)	58.77% (3.65%)	58.76% (4.76%)
pdc	0.055369	0.003998	47.87% (3.42%)	47.49% (1.08%)	46.99% (3.53%)	46.61% (0.97%)	60.06% (1.97%)	59.65% (2.71%)
tseng	0.006299	0.000545	62.39% (3.37%)	62.39% (0.06%)	61.44% (3.53%)	61.45% (0.03%)	73.29% (1.55%)	73.34% (0.39%)
dsip	0.049421	0.000842	55.71% (0.97%)	55.67% (0.72%)	55.50% (0.97%)	55.46% (0.71%)	68.00% (0.97%)	68.12% (1.38%)
diffeq	0.004697	0.000741	61.92% (2.93%)	61.93% (0.14%)	60.72% (3.13%)	60.73% (0.05%)	69.49% (1.64%)	69.57% (0.68%)
s298	0.010679	0.000884	57.06% (4.80%)	57.04% (0.29%)	56.26% (5.06%)	56.23% (0.25%)	66.74% (1.70%)	66.86% (0.78%)
bigkey	0.049461	0.000956	48.14% (0.98%)	47.84% (3.07%)	47.90% (0.97%)	47.60% (3.03%)	60.68% (1.57%)	60.54% (4.91%)
elliptic	0.015046	0.002065	62.71% (2.91%)	62.72% (0.01%)	61.04% (3.11%)	61.04% (0.01%)	74.94% (1.43%)	74.95% (0.04%)
frisc	0.012426	0.002612	64.32% (3.78%)	64.32% (0.01%)	62.05% (4.18%)	62.05% (0.00%)	75.11% (1.87%)	75.11% (0.05%)
s38584.1	0.058038	0.002767	62.00% (1.69%)	62.04% (0.09%)	61.44% (1.71%)	61.48% (0.09%)	73.89% (1.35%)	73.86% (0.25%)
s38417	0.056849	0.003581	54.86% (1.98%)	54.69% (1.47%)	54.07% (1.99%)	53.91% (1.43%)	67.36% (1.79%)	67.06% (2.15%)
clma	0.070958	0.008078	57.55% (4.70%)	57.39% (0.50%)	56.68% (5.06%)	56.53% (0.41%)	65.17% (1.51%)	64.93% (1.32%)
Ave.	0.037191	0.001947	52.71% (3.35%)	52.52% (1.03%)	52.03% (3.45%)	51.83% (0.96%)	62.71% (2.23%)	62.52% (2.57%)

Table 3: Total power achieved by EdTLC-LP and EdTLC-NW

[3] J. Lamoureux and S. J. Wilton, "On the interaction between power-aware FPGA CAD algorithms," in *Proc. Intl. Conf. Computer-Aided Design*, pp. 701–708, November 2003.

[4] 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.

[5] 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.

[6] F. Li, Y. Lin, and L. He, "FPGA power reduction using configurable dual-vdd," in *Proc. Design Automation Conf.*, June 2004.

[7] 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.

[8] Fei Li and Yan Lin and Lei He, "Vdd programmability to reduce FPGA interconnect power," in *Proc. Intl. Conf. Computer-Aided Design*, November 2004.

[9] Jason H. Anderson and Farid N. Najm, "Low-power programmable routing circuitry for FPGAs," in *Proc. Intl. Conf. Computer-Aided Design*, November 2004.

[10] Y. Lin and L. He, "Leakage efficient chip-level dual-vdd assignment with time slack allocation for FPGA power reduction," in *Proc. Design Automation Conf.*, June 2005.

[11] 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.

[12] Y. Hu, Y. Lin, L. He, and T. Tuan, "Simultaneous time slack budgeting and retiming for dual-vdd FPGA power reduction," in *Proc. Design Automation Conf.*, July 2006.

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

[14] S. C. Soheil Ghiasi, Elaheh Bozorgzadeh and M. Sarrafzadeh, "A unified theory of timing budget management," in *Proc. Intl. Conf. Computer-Aided Design*, November 2004.

[15] A. V. Goldberg, "An efficient implentation of a scaling minimum-cost flow algorithm," *Journal of Algorithms*, vol. 22, pp. 1–29, 1997.

[16] C. L. R. Rivest, T. Cormen, *An Introduction to Algorithms*. MIT Press, 1990.

[17] S. Yang, "Logic synthesis and optimization benchmarks, version 3.0," tech. rep., Microelectronics Center of North Carolina (MCNC), 1991.

[18] D. Lewis and et al, "The stratix routing and logic architecture," in *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, Feb 2003.

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

[20] "Xilinx product datasheets," in *http://www.xilinx.com/literature*.