Simultaneous Time Slack Budgeting and Retiming for Dual-Vdd FPGA Power Reduction *

Yu Hu[1], Yan Lin[1], Lei He[1] and Tim Tuan[2]
Electrical Engineering Dept., UCLA, Los Angeles, CA 90095[1]
Xilinx Research Lab., 2100 Logic Dr., San Jose, CA 95124[2]

ABSTRACT

Field programmable dual-Vdd interconnects are effective to reduce FPGA power. Assuming uniform length interconnects, existing work has developed time slack budgeting to minimize power based on estimating the lower bound of power reduction using dual-Vdd for given time slack. In this paper, we show that such lower bound estimation cannot be extended to mixed length interconnects that are used in modern FPGAs. We develop a technique to estimate power reduction using dual-Vdd for mixed length interconnects, and apply linear programming (LP) to solve slack budgeting to minimize power for mixed length interconnects. Experiments show 53% power reduction on average compared to single-Vdd interconnects. Furthermore, this paper presents a simultaneous retiming and slack budgeting algorithm to reduce power in dual-Vdd FPGAs considering placement and flip-flop binding constraints. The algorithm is based on mixed integer and linear programming (MILP) and achieves up to 20% power reduction compared to retiming followed by slack budgeting. We propose a runtime efficient flow to apply simultaneous retiming and slack budgeting only when it is necessary. To the best of our knowledge, this paper is the first in-depth study of simultaneous retiming and slack budgeting for dual-Vdd programmable FPGA power reduction while considering layout constraints.

Categories and Subject Descriptors: B.7.2[Hardware]: Integrated circuits – Design aids

General Terms: Algorithms, design

Keywords: Low power, retiming, FPGA

1. INTRODUCTION

Field programmable dual-Vdd techniques have been used for FPGA power reduction, e.g. [1, 2]. In this paper, we assume dual-Vdd interconnects identical to those in [3]. The interconnects consist of buffered wire segments. Buffers have programmable Vdd levels. There are Vdd-level converters at the inputs and outputs of logic blocks, but not between wire segments. Budgeting slacks in interconnects to minimize power is presented in [4, 3]. [3] is able to reduce more power than [4], where Vdd is defined for a routing tree. Uniform wire length and buffer size is assumed in [3]. However, the state-of-the-art commercial FPGAs have used wire segments of different lengths to improve performance [5]. Because the lower bound estimation of power reduction in [3] is no longer valid for mixed length interconnects, the first contribution of this paper is to develop a linear programming (LP) based slack budgeting for mixed-length interconnects based on an upper bound estimation of power reduction. The experimental results show 53% power reduction on average compared to single-Vdd interconnects.

The slack budgeting in [3] is applied only within combinational subcircuits. Simultaneously considering all combinational subcircuits in a sequential circuit may reduce more power, as illustrated in Figure 1, where circuits in (a) and (b) have the same clock period of 4 units. To change a buffer from VddH to VddL, one needs a slack of 2 units, no extra buffer can be powered by VddL in (a), but one extra buffer can be powered by VddL in (b). Because (b) can be obtained from (a) by retiming, simultaneous retiming and slack budgeting is able to reduce more power than slack budgeting alone in [3]. The second contribution of this paper is to develop mixed integer and linear programming (MILP) based simultaneous retiming and slack budgeting for power reduction while considering placement and flip-flop (FF) binding constraints. Note that retiming has been studied only for performance and area optimization in FPGAs [9, 10] without considering FF binding. Note that our retiming approach will not increase chip area since we only use available FF slots. To the best of our knowledge, this paper is the first in-depth study of simultaneous retiming and slack budgeting for dual-Vdd programmable FPGA power reduction while considering layout constraints including FF binding.

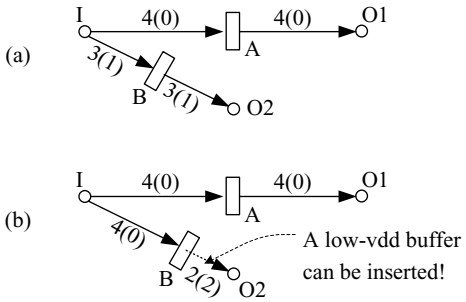

Figure 1: Retiming followed by budgeting vs. Simultaneous approach

The rest of this paper is organized as follows. Section 2 introduces background and modeling. Section 3 describes a dual-Vdd

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

DAC 2006, July 24–28, 2006, San Francisco, California, USA.

slack budgeting algorithm for mixed wire length FPGAs. Section 4 presents a post-layout stage retiming algorithm. Experimental results are given in Section 5. Section 6 concludes the 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.

$\mathcal{G}(\mathcal{V},\mathcal{E})$	retiming 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_e	FF number in $e(i,j)$
r_u	retiming value in node u

Table 1: Notations frequently used in this paper

The Elmore delay model is used to calculate the routing delay. Following [3], we define the *fanout cone* of a switch as the sub-tree of the routing tree rooted at the switch. To incorporate dual-Vdd into 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 [6], and such impact is ignored in this paper.

There are three types of power sources in FPGAs, switching power, short-circuit power and static (leakage) power. The first two contribute to the dynamic power and can only occur when a signal transition happens at the gate output. Although a timing change may change the transition density, we assume (as in [3]) that the transition density for an interconnect switch will not change when VddL is used. 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 unused switches are power-gated. Let v_{ij} indicate Vdd-level of the j^{th} switch in $\mathcal{R}_i$ as follows.

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

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 the j^{th} switch in the 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.

2.2 Retiming with FF constraints

A directed retiming graph is constructed to model the circuit for sequential timing analysis. Vertices represent the inputs/outputs of basic circuit elements such as 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. Each edge is annotated with the delay $d(e)$ required to pass through the circuit element or routing and the weight $w(e)$, which is the number of FFs inserted in it. We use $\mathcal{PI}$ and $\mathcal{PO}$ to represent the set of primary inputs and outputs, respectively.

In order to retime an FPGA design during post-layout optimization, we need to consider placement and FF binding constraints. In older FPGA devices, a logic cell (LC) has only one output. If an LUT drives an FF within the same LC, the combinational output of an LUT is NOT allowed to drive other FFs since it cannot be seen by other FFs. In newer devices such as the Xilinx Virtex-II [7] or Altera Stratix II [8], an LC has both the combinational output (without FF) and sequential output (with FF). If an LUT drives an FF within the same LC, the sequential output can still drive other FFs either within the same cluster (through local routing) or in other clusters (through global routing). In other words, FFs can be cascaded. FFs within a cluster can be cascaded through an LUT (acting as a WIRE) or a MUX. FFs within different clusters can be cascaded going through a global routing and an LUT or MUX in the other cluster. In post-layout retiming, we try to keep most layout (both placement and global routing) unchanged, therefore we allow an LUT to drive any available FFs within the same cluster, but not FFs in a different cluster.

3. DUAL-VDD SLACK BUDGETING

In this section, we present an LP-based time slack budgeting algorithm for mixed interconnect wire lengths. 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. Similar to the LP based algorithm for uniform interconnect wire length in [3], we then perform a bottom-up assignment algorithm to achieve the optimal solution within each routing tree for the allocated time slack, and finally perform a refinement step to leverage surplus time slack.

3.1 Estimation of Interconnect Power Reduction

Estimating power reduction for the allocated slack is the key to enable the LP based algorithm. The slack S_{ij} of a connection between the source and the 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}. 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 greater than the upper bound cannot lead to more VddL switches. We define the *useful slack* of each routing tree sink as the slack less than this upper bound. For the rest of the paper, we use slack to represent the useful slack. The slack upper bound constraints can be expressed as

$$0 \le S_{ik} \le D_{ik} \qquad 0 \le i < N_r \wedge 1 \le k \le N_k(i) \qquad (2)$$

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 the k^{th} sink in $\mathcal{R}_i$ when VddL is assigned to all the switches in that path.

Given a routing tree with arbitrary topology and allocated slack for each sink, we need to estimate the power reduction that can be achieved. We use l_{ik} to represent the number of switches in the path from the source to the k^{th} sink in $\mathcal{R}_i$. We first transform slack

S_{ik} into s_{ik}, which is expressed in number of switches as follows.

$$s_{ik} = \frac{S_{ik}}{D_{ik}} \cdot l_{ik} \tag{3}$$

We then estimate the number of VddL switches that can be achieved using s_{ik}. Let c_{ij} represent the load capacitance of the j^{th} switch in $\mathcal{R}_i$, and C_{ik} represent the total load capacitance of the switches in the path from the source to the k^{th} sink in $\mathcal{R}_i$. We define *sink list* $\mathcal{SL}_{ij}$ as the set of sinks in the fanout cone of the j^{th} switch in $\mathcal{R}_i$. [3] presents the lower bound of the number of VddL switches for the allocated slack in uniform length cases as

$$F_n^{low}(i) = \sum_{j=0}^{N_s(i)-1} min(\frac{s_{ik}}{C_{ik}} \cdot c_{ij} : \forall k \in \mathcal{SL}_{ij}) \tag{4}$$

We find that $F_n^{low}(i)$ is not a lower bound in mixed length cases, where the size of switches along a source-to-sink path may be different. Figure 2 shows a source-to-sink path, which includes two switches with different sizes. They need 1.5 and 0.5 slacks to be powered by VddL, respectively. If we assign 1.9 slack in the sink, $F_n^{low}(i) = 1.9$ based on (4). However, when we perform bottom-up Vdd assignment, the lower stream switch consumes 0.5 slack and there is 1.4 slack left for the upper stream switch, which is not enough for it to be VddL. Therefore, we get only one VddL switch with used slack of 0.5 instead of the estimated 1.9.

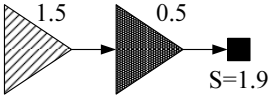

Figure 2: $F_n^{low}(i)$ in a mixed length case

We propose an upper bound $F_n^{up}(i)$ of VddL switch number in $\mathcal{R}_i$ by summing up all $s_{ik}c_{ij}/C_{ik}$ in its fanout cone as the slack distributed to the switch. It is expressed as

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

We then estimate the power reduction for $\mathcal{R}_i$. The upper bound $P_{dr}^{up}(i)$ of dynamic power reduction of the tree $\mathcal{R}_i$ is estimated as the sum of the dynamic power reduction of each switch in $\mathcal{R}_i$ and can be expressed as

$$P_{dr}^{up}(i) = 0.5 f_{clk} \cdot \Delta Vdd^2 \sum_{j=0}^{N_s(i)-1} F_n^{up}(i) \cdot f_s(i,j) \cdot c_{ij} \tag{6}$$

Similarly, the upper bound $P_{lr}^{up}(i)$ of leakage power reduction of $\mathcal{R}_i$ is the sum of the leakage power reduction of each switch in $\mathcal{R}_i$ and can be expressed as

$$P_{lr}^{up}(i) = \sum_{j=0}^{N_s(i)-1} F_n^{up}(i) \cdot \Delta P_s(i,j) \tag{7}$$

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

In the experiments, we find that the power estimation (4) developed in [3] does not work well in simultaneous retiming and slack budgeting (Section 4). Note that the upper bound based estimation gives more weight to the shared switches[1], and the shared switches often have larger transit density than non-shared ones, which means that more power gain can be obtained by assigning the shared switches to VddL.

[1] A shared switch is a switch which has more than one sink in its fanout cone

3.2 LP Problem Formulation

To formulate budgeting as a mathematical programming problem, we need to explicitly express the constraints and objective function. In this problem, 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

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

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

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

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 function is to maximize the estimation of interconnect power reduction given by (6) and (7). It can be expressed as

$$Maximize \sum_{i=0}^{N_r-1} 0.5 f_{clk} \Delta Vdd^2 F_n^{up}(i) f_s(i,j) c_{ij} + \sum_{i=0}^{N_r-1} F_n^{up}(i) \Delta P_s(i,j) \tag{11}$$

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

$$S_{ik} = a(p_{ik}) - a(p_{i0}) - d(p_{i0}, p_{ik}) \quad 0 \le i < N_r \wedge \forall p_{ik} \in \mathcal{FO}_{p_{i0}} \tag{12}$$

where vertex p_{i0} is the source of $\mathcal{R}_i$ in $\mathcal{G}$, vertex p_{ik} is the k^{th} sink of $\mathcal{R}_i$ in $\mathcal{G}$, S_{ik} is the slack allocated to the 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

$$a(u) + d(u,v) \le a(v) \qquad \forall u \in \mathcal{V} \wedge u \notin \mathcal{SRC} \wedge v \in \mathcal{FO}_u \tag{13}$$

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 (11), slack upper bound constraints (2), and timing constraints (8), (9), (12) and (13). It is easy to verify that all the constraints are linear, and the objective function (11) is also linear. Hence we have the following theorem.

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

Time slack is allocated to each routing tree by solving the time slack allocation problem. Then the net-level bottom-up assignment and refinement from [3] are modified to consider mixed-length interconnects and to leverage the allocated slack. Note that our upper bound based power estimation may give an overestimation of power reduction and the number of VddL switches, and the net-level bottom-up Vdd assignment guarantees the legalization of final solutions.

4. POST-LAYOUT RETIMING

4.1 Retiming and Delay Constraints

The MILP formulation for retiming synchronous circuits is originally presented in [11] to minimize clock period. We extend the MILP formulation to consider interconnect delay and get the following theorem

Theorem 2. Let $G = (V, E, d, w)$ be a synchronous circuit, and let c be a positive real number. Then there exists a retiming r of G such that $\Phi(G_r) \leq c$ if and only if there exists an assignment of real values $a(v)$ and an integer value $r(v)$ to each vertex $v \in V$ such that the following conditions are satisfied:

$$-a(v) \leq -\max_{u \in Fanin(v)} d(u,v), \quad \forall v \in V \tag{14}$$

$$a(v) \leq c, \quad \forall v \in V \tag{15}$$

$$r(u) - r(v) \leq w(e), \quad \forall e(u,v) \in E \tag{16}$$

$$a(u) - a(v) \leq -d(u,v), \quad \forall e(u,v) \text{ s.t. } r(u) - r(v) = w(e) \tag{17}$$

where G_r is the retimed circuit and $\Phi(G_r)$ is the clock period of G_r.

Suppose $R(v) = r(v) + a(v)/c$, then the above retiming constraints can be re-written as

$$r(v) - R(v) \leq -\max_{u \in Fanin(v)} d(u,v)/c, \quad \forall v \in V \tag{18}$$

$$R(v) - r(v) \leq 1, \quad \forall v \in V \tag{19}$$

$$r(u) - r(v) \leq w(e), \quad \forall e(u,v) \in E \tag{20}$$

$$R(u) - R(v) \leq w(e) - d(u,v)/c, \quad \forall e(u,v) \in E \tag{21}$$

In the next two subsections, we propose additional constraints to consider placement and FF binding constraints.

4.2 FF Number Constraints

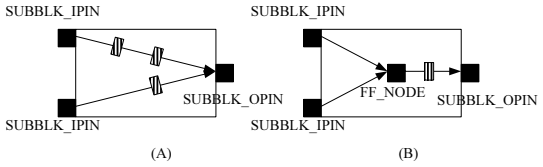

Figure 3: A dummy node to consider FF sharing

As described in Section 2.2, delay values are associated with timing edges. An LC is then represented by several nodes and edges in the retiming graph (see Figure 3). (A) connects the inputs of the LUT to the output of the LC directly in the retiming graph, the FF number on each of such edges may be different after retiming, which leads to an illegal FF assignment in the FPGA as the FF number in each input-output pair of an LC should be the same with a legal retiming. To tackle this problem, we add a dummy FF node to each LC (including the LC originally in combinational mode), and constrain that FFs can only be inserted at edges from FF node to LC output node as shown in (B). To ensure an LUT drives only FFs within the same cluster after retiming, the total FFs used within one cluster can not exceed the number of available FFs in the cluster. Therefore, we have the following constraints

$$\sum_{e(u,v) \in E_{Ci}} w(e) + r(v) - r(u) \leq S_{Ci}, \quad \forall C_i \tag{22}$$

$$r(v) - r(u) = 0, \quad \forall e(u,v) \in E/E_{ff} \tag{23}$$

where set E_{ff} comprises all edges from FF nodes to LC output nodes, set E_{Ci} comprises all E_{ff} edges in cluster i, and S_{Ci} denotes the FF number in cluster i.

4.3 Consider FF Delay

There are two runtime modes of an LC, i.e. combinational and sequential modes. Figure 4 shows LC input delay (from LC input to FF node) and LC output delay (from FF node to LC output). When the LUT in an LC drives a least one FF, it runs in a sequential mode. The LC input delay is T_{seq_in} and its output delay is T_{seq_out}. Otherwise, it works in a combinational mode, and the LC input and output delays are T_{comb} and 0, respectively.

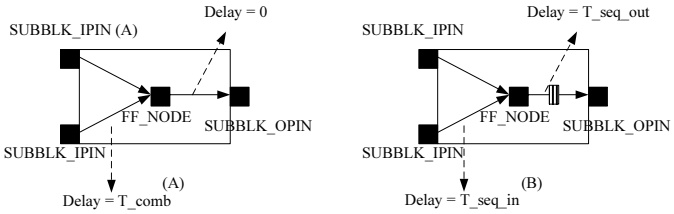

Figure 4: (a)Combinational mode of LC (b) Sequential mode of LC

Formally, suppose the LC input and output delay are D_{in} and D_{out}, respectively. Then we have

$$D_{in} = \begin{cases} T_{seq\_in}, & \text{if } w'(e) > 0 \\ T_{comb}, & \text{if } w'(e) = 0 \end{cases} \tag{24}$$

$$D_{out} = \begin{cases} T_{seq\_out}, & \text{if } w'(e) > 0 \\ 0, & \text{if } w'(e) = 0 \end{cases} \tag{25}$$

where $w'(e) = w(e) + r(v) - r(u)$ is the FF number in edge $e(u,v)$ after retiming [11].

We can write these as linear expressions as follows.

$$T_{comb} \leq D_{in} \leq T_{seq\_in} \tag{26}$$

$$D_{in} \geq T_{seq\_in} \cdot w'(e) \tag{27}$$

$$0 \leq D_{out} \leq T_{seq\_out} \tag{28}$$

$$D_{out} \geq T_{seq\_out} \cdot w'(e) \tag{29}$$

We incorporate these constraints into our MILP formulation, and rewrite (18) and (21) for timing edges within an LC as follows.

$$r(v) - R(v) \leq -D_{in,out}/c, \quad \forall e(u,v) \in E_{in,out} \tag{30}$$

$$R(u) - R(v) \leq w(e) - D_{in,out}/c, \quad \forall e(u,v) \in E_{in,out} \tag{31}$$

where E_{in} and E_{out} are the LC input and LC output edge sets, respectively.

4.4 Clock Period Minimization Retiming

[11] has proved that the minimal possible clock period is equal to the delay of a certain path of the circuit, which enables us to find the minimal clock period by performing binary search among all path delay values. To test whether each potential clock period c is feasible, we formulate the *post-stage clock period minimization retiming problem* using a constant objective function, and retiming and delay constraints (18), (19), (20), (21), (30), (31), and FF number constraints (23), (22). It is easy to verify that all the constraints are linear. Hence we have the following theorem.

Theorem 3. The *post-stage clock period minimization retiming problem* is a mixed integer linear programming (MILP) problem.

4.5 Simultaneous Retiming and Budgeting

The following important observation enables us to consider time slack explicitly in our formulation.

OBSERVATION 1. *The real value $a(v)$ assigned in node v in Theorem 2 is its arrival time after retiming.*

Based on this observation, the time slack in edge $e(u, v)$ can be expressed as

$$\begin{aligned} S(u,v) &= a(v) - a(u) - d(u,v) \\ &= [R(v) - R(u) + r(u) - r(v)] \cdot c - d(u,v) \end{aligned} \quad (32)$$

Therefore, for the edges corresponding to routing in $\mathcal{G}$, the constraints (12) considering slack can be rewritten as

$$\begin{aligned} S_{ik} &= [R(p_{ik}) - R(p_{iO}) + r(p_{iO}) - r(p_{ik})] \cdot c - d(p_{i0}, p_{ik}) \\ & 0 \le i < N_r \wedge \forall p_{ik} \in \mathcal{FO}_{p_{i0}} \end{aligned} \quad (33)$$

If we substitute S_{ik} in (3) with (33), and use the power estimation (11) in Section 3 as the objective function, and employ all constraints in the *post-stage clock period minimization retiming problem* plus the slack bound constraints (2), we can consider timing and slack budgeting simultaneously. It is easy to verify that all the constraints and objective functions are linear. Hence we have the following theorem.

Theorem 4. The *post-stage simultaneous retiming and slack budgeting problem* is a mixed integer linear programming (MILP) problem.

5. EXPERIMENTS AND DISCUSSION

We conduct the experiments on the 10 sequential circuits in the MCNC benchmark [12]. We map them into a FPGA with LUT size of 4 and cluster size of 10. We use the same Vdd-programmable logic blocks and interconnects in [3], 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 trade-off, as suggested in [5]. We use 25x and 10x switches to drive length 8 and length 4 wires, respectively. The parameters of switches are extracted by SPICE simulation. The unused interconnect switches are power-gated in all cases. Similar to [13], we customize the FPGA chip size for each benchmark circuit and use the smallest chip that fits each benchmark. Considering the VddL/VddH ratio between 0.6 $\sim$ 0.7 suggested in [14], we use 1.3v for VddH and 0.8v for VddL in our experiments at $100nm$ technology node. We use mosek [15] to solve the LP and MILP problem. The experimental data are collected on a linux workstation with a $1.9GHz$ Xeon CPU and $2GB$ memory.

We first use VPR [16] for single-Vdd placement and routing. Before applying retiming and budgeting algorithms to the Vdd-programmable interconnects, a sensitivity based assignment [1] is first performed to assign Vdd-levels for Vdd-programmable logic blocks without performance loss[2]. After Vdd assignment for clusters, we perform clock period minimization retiming followed by slack budgeting and Vdd assignment, namely *CRT+SB*. Alternatively, we perform simultaneous retiming and slack budgeting followed by Vdd assignment, namely *RTSB*, while keeping the minimal clock period achieved by *CRT+SB*.

5.1 Experimental Results

We run our LP based budgeting algorithm presented in Section 3 for uniform length cases, and compare the results with [3], which is shown in Table 2. We find that our algorithm obtains similar power and runtime as [3]. Table 3 shows the experimental results for the LP based slack budgeting algorithm proposed in Section 3 for mixed-length interconnects. Column *svdd* shows the power dissipation (10^{-3}W) of the solutions produced by the algorithm with single Vdd and power-gating. Columns under $dual - Vdd$ show the the percentage of VddL switches (column $VddL$), total power dissipation (column *power*), power reduction (in parenthesis under column *power*) on *svdd*, clock period (column *clock* in ns) and the total runtime[3] (column *time* in s). Our algorithm can achieve 85% VddL assignment and 53% power reduction for mixed length interconnect wire on average, respectively.

	svdd	[3]		dual-Vdd	
circuit		power	time	power	time
tseng	6.8	2.6(-62%)	17	2.6(-62%)	22
dsip	50.3	22.3(-56%)	32	22.3(-56%)	37
diffeq	5.4	2.1(-62%)	52	2.1(-62%)	55
s298	11.6	5.0(-57%)	83	5.0(-57%)	82
bigkey	50.4	26.1(-48%)	68	26.3(-48%)	68
elliptic	17.1	6.4(-63%)	185	6.4(-63%)	185
frisc	15.0	5.4(-64%)	297	5.4(-64%)	309
s38584.1	60.8	23.1(-62%)	345	23.1(-62%)	325
s38417	60.4	27.3(-55%)	609	27.4(-55%)	621
clma	79.0	33.6(-58%)	1934	33.7(-57%)	1973
ave	35.7	15.4(-59%)	363	15.4(-59%)	368

Table 2: Uniform-length slack budget

		svdd	dual-Vdd			
Circuit	Cluster#	power	VddL	power	clock	time
tseng	131	10.2	96%	3.9(-62%)	13.1	32
dsip	162	111.4	72%	66.5(-40%)	6.2	44
diffeq	195	10.2	88%	4.0(-61%)	13.5	41
s298	256	23.3	83%	10.9(-53%)	24.3	96
bigkey	294	96.1	73%	57.9(-40%)	7.0	89
elliptic	421	35.9	90%	14.9(-58%)	17.3	229
frisc	595	29.6	98%	11.1(-63%)	23.7	479
s38584.1	704	121.1	92%	51.0(-58%)	11.9	421
s38417	847	125.8	82%	67.2(-47%)	15.7	709
clma	1358	155.4	75%	76.9(-51%)	23.2	2735
ave	496	77.7	85%	36.4(-53%)	15.5	488

Table 3: Mixed-length slack budget

Table 4 shows retiming results by *CRT+SB* and *RTSB* for mixed-length interconnect, respectively. *CRT+SB* performs a binary search to find the minimal clock period achieved by retiming (column *clock*). To control runtime in *CRT+SB*, we use $1\%T_{spec}$ as the searching step, and search possible clock period within $[90\%T_{spec}, T_{spec}]$, where T_{spec} is the clock period after LP based slack budgeting, as we find that the clock period has been well optimized in the post-layout stage. For *RTSB*, the percentage of VddL switches, total power dissipation (10^{-3}W) and power reduction (in the brackets) compared to *CRT+SB* are shown in Table 4. The minimal clock periods by *CRT+SB* are also given in this table. Note that *RTSB* can achieve up to 20% power reduction compared to *CRT+SB*, while negligible power reduction is obtained overall, as there is not much room left for retiming and further power reduction. In the next subsection, we propose an efficient design flow for retiming and Vdd assignment in the post-layout stage based on our experimental results.

Table 5 gives the runtime of *RTSB* and *CRT+SB*. For each of the given clock periods, we only need to test the feasibility of the MILP, which takes relatively shorter runtime than solving an MILP. When a feasible minimal clock period is found, we solve the MILP. Therefore, the total runtime for *CRT+SB* is longer than *RTSB* on average, due to the cost of extra feasibility testing. The runtime of *RTSB* is not consistently monotonic with respect to the size of

[2] We do consider the fact that VddL logic blocks consume time slack.

[3] The total runtime includes the runtime consumed by the placement and routing with VPR, slack budgeting and Vdd assignment.

the circuit, e.g. the runtime of $frisc$ (595 CLBs) by $RTSB$ is 3x of $s38417$ (847 CLBs), as the runtime to solve a MILP problem is sensitive to the structure of the problem.

5.2 Discussion and Overall Tool Flow

Figure 5 shows a clear view of the factors that influence the gain of $RTSB$. The percentage power reduction before and after refinement, and the percentage clock period reduction are shown in this figure.

For each benchmark circuit, the improvement obtained by retiming is highly dependent on the distribution of the critical path. We find that only the clock period of circuit $tseng$ and $bigkey$ can be reduced by $CRT+SB$, which indicates that there is not much room left for retiming in the post-stage optimization. Nevertheless, $RTSB$ can further reduce power while keeping the minimal clock period. As mentioned in Section 3, a refinement process is performed to leverage surplus time slack. The curve "power before refinement" in Figure 5 shows the percentage of power reduction before refinement. We can achieve 8% power reduction on $CRT+SB$ before refinement. We also find that the power reduction by $RTSB$ is reduced due to refinement. Note that the slack values in our formulation are continuous variables instead of discrete ones in reality, which leads to some unusable slacks. The refinement assigns VddL using available slack locally. After refinement, $RTSB$ can still achieve up to 20% more power savings ($bigkey$) compared to $CRT+SB$.

We find from Table 4 and Figure 5 that VddL percentage and clock period reduction by $CRT+SB$ are good indicators for $RTSB$. For those circuits which already have high VddL percentages, there is not much room left for further power reduction. On the other hand, $RTSB$ is expected to achieve more power reduction if $CRT+SB$ can get more clock period reduction. Therefore, an efficient design flow in post-layout Vdd assignment and retiming is as follows. We perform $CRT+SB$. Then we only choose those circuits that get relatively large clock period reduction and small VddL percentage to perform $RTSB$. As mentioned in Section 3, $RTSB$ may give an overestimation of power reduction, and produces unusable slacks, which may lead to even a greater power than $CRT+SB$ ($diffeq$ and $s38584.1$). In these cases, we just keep the results produced by $CRT+SB$.

	CRT+SB			RTSB	
circuit	VddL	power	clock	VddL	power
tseng	94%	4.6	13.0(-1%)	96%	4.5(-1%)
dsip	72%	66.5	6.2(0%)	71%	66.5(0%)
diffeq	88%	4.0	13.5(0%)	86%	4.1(1%)
s298	83%	10.9	24.3(0%)	83%	10.8(-1%)
bigkey	73%	57.8	6.8(-2%)	79%	46.1(-20%)
elliptic	90%	14.9	17.3(0%)	90%	14.9(0%)
frisc	98%	11.1	23.7(0%)	98%	11.1(0%)
s38584.1	92%	50.7	11.9(0%)	90%	51.0(1%)
s38417	82%	66.7	15.7(0%)	82%	66.1(-1%)
clma	75%	76.9	23.2(0%)	75%	74.4(-3%)
	85%	36.4	15.5(-0.2%)	85%	34.9(-2%)

Table 4: Retiming for mixed-length interconnects

	CRT+SB		RTSB
circuit	total	binarySearch	total
tseng	43	4	44
dsip	74	9	66
diffeq	192	124	85
s298	654	470	230
bigkey	302	77	254
elliptic	575	145	473
frisc	1286	171	2402
s38584	1072	193	1092
s38417	1687	1029	863
clma	9736	2427	8580
ave	1562	465	1409

Table 5: Runtime (seconds) for retiming algorithms

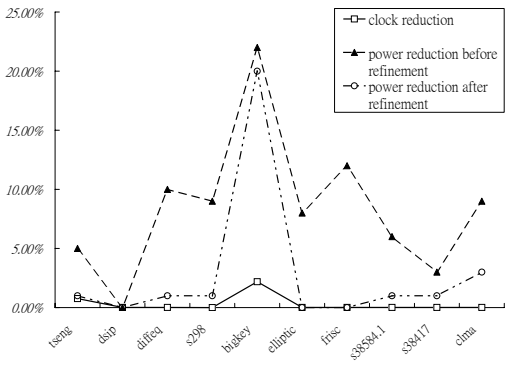

Figure 5: Power and clock period reduction

6. CONCLUSION

To reduce power in dual-Vdd FPGAs, we have presented a linear programming (LP) based time slack allocation algorithm for mixed interconnect wire lengths. Experiments show that our algorithm obtains similar power and runtime as [3] for dual-Vdd uniform interconnects, and reduces power by 53% compared to single-Vdd mixed length interconnects. We have also proposed a simultaneous retiming and slack budgeting formulation considering post-layout constraints, and reduced up to 20% more power than retiming followed by slack budgeting. Finally, we have proposed a runtime efficient flow to apply simultaneous retiming and slack budgeting only when it is necessary.

7. REFERENCES

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

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

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

[4] Y. Lin, F. Li, and L. He, "Power modeling and architecture evaluation for FPGA with novel circuits for vdd programmability," in *DAC*, Februray 2005.

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

[6] Y. Lin and L. He, "Dual-vdd interconnect with chip-level time slack allocation for fpga power reduction," *TCAD*, 2006.

[7] Xilinx Corporation, "Virtex-II 1.5V Platform FPGA Complete Data Sheet", Jun 2002.

[8] D. Lewis et al, "The stratix ii routing and logic architecture," in *FPGA*, Feb 2005.

[9] C.-Y. Ych and M. Marek-Sadowska, "Delay budgeting in sequential circuit with application on fpga placement," in *DAC*, Jun 2003.

[10] C.-Y. Ych and M. Marek-Sadowska, "Minimum-area sequential budgeting for fpga," in *ICCAD*, Nov 2003.

[11] C.E.Leiserson and J.B.Saxe, "Retiming synchronous circuitry," *Algorithmica*, pp. 5–35, 1991.

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

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

[14] M. Hamada et al, "A top-down low power design technique using clustered voltage scaling with variable supply-voltage scheme," in *IEEE CICC*, pp. 495–498, 1998.

[15] http://www.mosek.com.

[16] V. Betz et al, *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, Feb 1999.