Field Programmability of Supply Voltages for FPGA Power Reduction

Fei Li, Yan Lin, *Student Member, IEEE*, and Lei He

***Abstract*—Power reduction is of growing importance for field-programmable gate arrays (FPGAs). In this paper, we apply programmable supply voltage (Vdd) to reduce FPGA power. We first design FPGA logic fabrics using dual-Vdd levels and show that field-programmable power supply is required to obtain a satisfactory power-versus-performance tradeoff. We further design FPGA interconnect fabrics for fine-grained Vdd programmability with minimal increase of the number of configuration static-random-access-memory cells. With a simple yet practical computer-aided design flow to leverage the field-programmable dual-Vdd logic and interconnect fabrics, we carry out a highly quantitative study using placed and routed benchmark circuits, and delay, power, and area models obtained from detailed circuit designs. Compared to single-Vdd FPGAs with the Vdd level suggested by the International Technology Roadmap for Semiconductors for 100-nm technology, field-programmable dual-Vdd FPGAs reduce the total power by 47.61% and the energy-delay product by 27.36%.**

***Index Terms*—Dual Vdd, field-programmable gate array (FPGA) architecture, power reduction, supply-voltage programmability.**

I. Introduction

A FIELD-PROGRAMMABLE gate array (FPGA) is an attractive design platform due to its low nonrecurring engineering (NRE) cost and short time to market. However, for a given register transfer level (RTL) design, it has a much lower power efficiency than the application-specific integrated circuit (ASIC) because a large number of transistors are used for field programmability. FPGA power modeling and analysis have drawn growing attentions. Poon *et al.* [1] and Li *et al.* [2] present flexible power models for parameterized FPGA architectures and show that both interconnect and leakage power are significant power components. Tuan and Lai [3] analyze the leakage power of a commercial FPGA architecture in 90-nm technology and quantifies the leakage power challenge for nanometer FPGAs. As power consumption becomes an increasingly important design constraint, FPGA power reduction has also been studied recently. Anderson *et al.* [4] introduces an inversion method to reprogram the FPGA configuration bits and reduce the leakage power of multiplexers (MUXs) without additional hardware cost. Lamoureux and Wilton [5] develop a suite of power-aware computer-aided design (CAD) algorithms for existing FPGA architectures. Other work involves designing power-efficient FPGA circuits. For example, Gayasen *et al.* [6] investigate the power gating of logic fabrics and apply region-constrained placement to reduce the leakage power of unused logic blocks.

Existing FPGAs usually use a uniform supply voltage (Vdd) for their array cores. One can scale down the Vdd level of an entire FPGA array to reduce power, but the power saving is obtained at the cost of performance degradation. To further improve power efficiency, we believe that different Vdd levels should be explored. A dual-Vdd technique applies high supply voltage (VddH) to devices on critical paths to maintain performance and low supply voltage (VddL) to devices on noncritical paths to reduce power. The dual-Vdd technique has been successfully applied in ASIC (e.g., [7]) and is able to achieve a better power-versus-performance tradeoff than Vdd scaling does. However, a predefined dual-Vdd FPGA fabric in general cannot achieve a better power-versus-performance tradeoff than Vdd scaling because its predefined dual-Vdd pattern is not flexible enough to accommodate a variety of applications [8]. Therefore, field programmability of the dual-Vdd pattern is a must to reduce FPGA power through dual-Vdd techniques.

The concept of Vdd programmability for FPGA was first introduced in [8], and the preliminary studies on Vdd-programmable logic and interconnect fabrics were reported in [9] and [10], respectively. Compared to papers [8]–[10], this paper presents more analysis and experimental results and also significantly reduces the circuit overhead to implement the Vdd programmability. To the best of our knowledge, this paper and [8]–[10] present the first in-depth study of field-programmable supply voltages for FPGAs. A variety of new developments on Vdd programmability in FPGAs have been introduced in recent papers such as [11]–[15] (to be discussed in Section VII). Dual-Vdd technology mapping and clustering were also studied in [16] and [17].

The rest of this paper is organized as follows. Section II presents the background knowledge and baseline architecture for comparison. Section III introduces the field-programmable power supply to logic fabrics and discusses the detailed circuit and fabric design. Section IV discusses the CAD flow with consideration of Vdd-programmable logic fabrics. Experimental results of logic and local interconnect power reduction are presented in Section V. Section VI further applies Vdd programmability to global interconnects (global interconnects and interconnect fabric are used interchangeably in this paper) and discusses the design of Vdd-programmable interconnect switches. We conclude this paper in Section VII.

Manuscript received March 31, 2005; revised September 20, 2005 and May 4, 2006. This work was supported by the NSF under CAREER Award CCR-0401682 and Grant CCR-0306682. This paper was recommended by Associate Editor M. D. F. Wong.

The authors are with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: lhe@ee.ucla.edu).

Digital Object Identifier 10.1109/TCAD.2006.884848

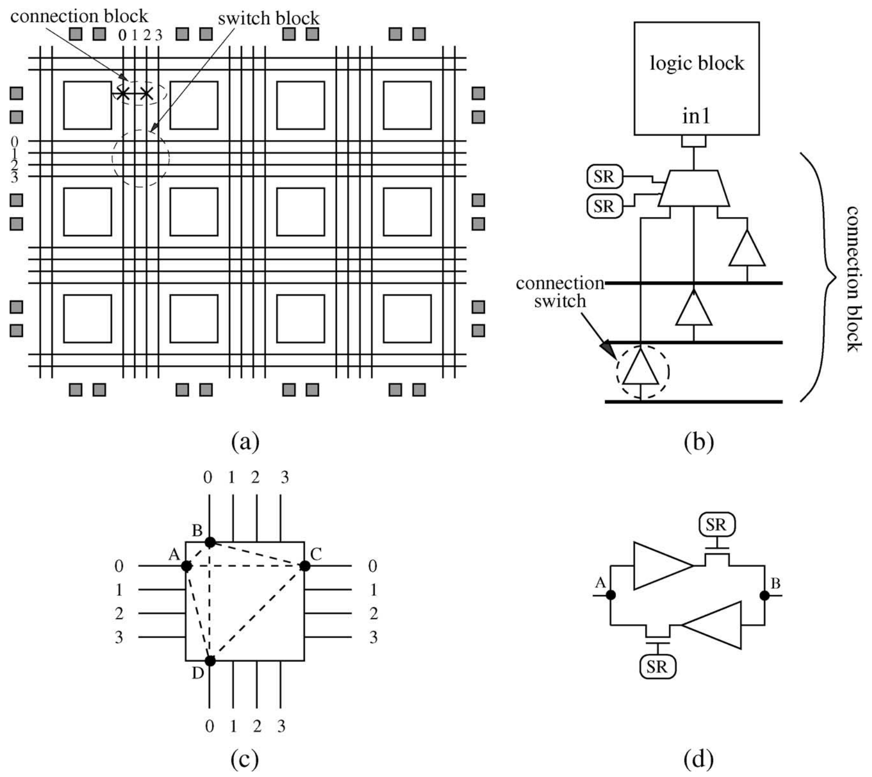


Fig. 1. (a) Island-style routing architecture. (b) Connection block and connection switch. (c) Switch block. (d) Routing switches.

II. Background and Preliminaries

A. Cluster-Based Island-Style FPGAs

This paper assumes the cluster-based island-style FPGA architecture from [18], which is shown in Fig. 1(a). Logic blocks are surrounded by programmable routing channels, and routing wires in both horizontal and vertical channels are segmented by "switch blocks." Fig. 1(c) shows a subset switch block [19], where the incoming track can only be connected to outgoing tracks with the same track number. The connections in a switch block [represented by the dashed lines in Fig. 1(c)] are programmable routing switches. As shown in Fig. 1(d), routing switches can be implemented by tristate buffers, and each connection needs two tristate buffers so that it can be programmed independently for either direction.[1] The wire segments in routing channels are connected to the input/output pins of a logic block by connection blocks, as shown in Fig. 1(b).

In this paper, we assume that a logic block contains ten basic logic elements (BLEs), with each BLE having one four-input lookup table (LUT) and one flip-flop, and the interconnect structure contains 100% tristate buffers (rather than a mix of buffers and pass transistors). We customize an FPGA array for each individual benchmark circuit so that the array size just fits the given circuit for logic cell placement. We decide the routing channel width W in the same way as the architecture study in [18], i.e., $W = 1.2W_{\min}$, where $W_{\min}$ is the minimum channel width required to route the given circuit successfully. This channel width W represents a "low-stress" routing situation that usually occurs in commercial FPGAs for "average" circuits. Similar to [18], we conduct experiments by placing and routing MCNC benchmarks in 100-nm technology.

[1]Modern FPGAs often use unidirectional routing switches. Because of the limitation of the VPR tool set [20], we assume bidirectional switches that are implemented by tristate buffers. However, our low-power techniques apply to unidirectional switches.

B. Low-Leakage Static RAM (SRAM)

FPGAs use a large number of SRAM cells for field programmability. The configuration SRAM cells are used either to program the logic function of LUTs or to configure the connections between routing wires. Previous work has shown that high threshold voltage (high Vt) can be used for transistors in SRAM cells to reduce SRAM leakage [3], [8]. Fig. 2 uses an LUT as an example to illustrate the application of such low-leakage technique. The entire LUT is partitioned into two different regions. All the configuration SRAM cells belong to region I, while the rest including MUX-tree and input buffers becomes region II. Note that the two regions are dc disconnected due to the inverters at the output of the SRAM cells. The content of the SRAM cells does not change after the LUT is configured, and the SRAM cells always stay in the READ status. Therefore, they only consume leakage power (excluding dynamically reconfigurable designs), and their READ or WRITE delays are irrelevant to the design performance. Ideally, we can increase Vt in region I as much as possible to achieve maximal leakage reduction without runtime delay penalty. In reality, a too high Vt increases the SRAM WRITE time (i.e., FPGA configuration time) significantly. In this paper, we increase the Vt of SRAM cells to obtain 15× SRAM leakage reduction, which only increases the configuration time by 13%. This tradeoff is justifiable because the configuration time is not critical in most FPGA applications. Note that the high-Vt

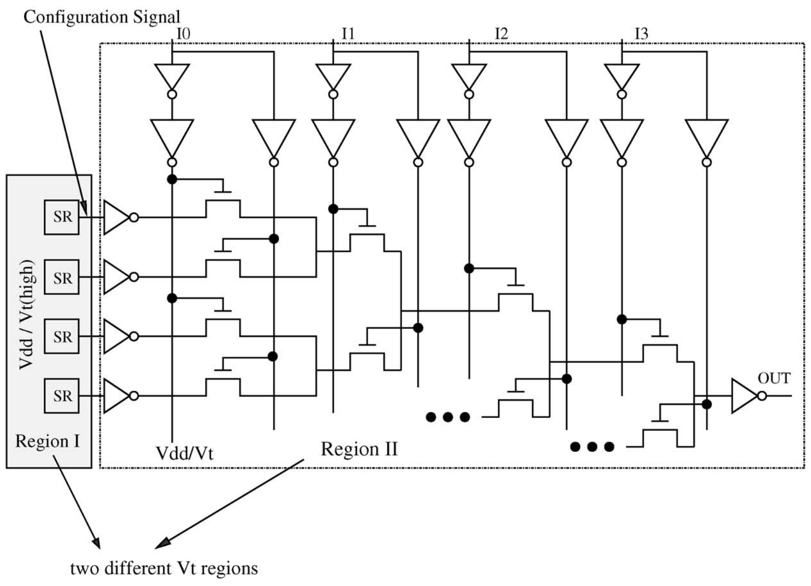


Fig. 2. Schematic of a four-input LUT using high-Vt (low-leakage) SRAM cells ("SR" stands for SRAM cell).

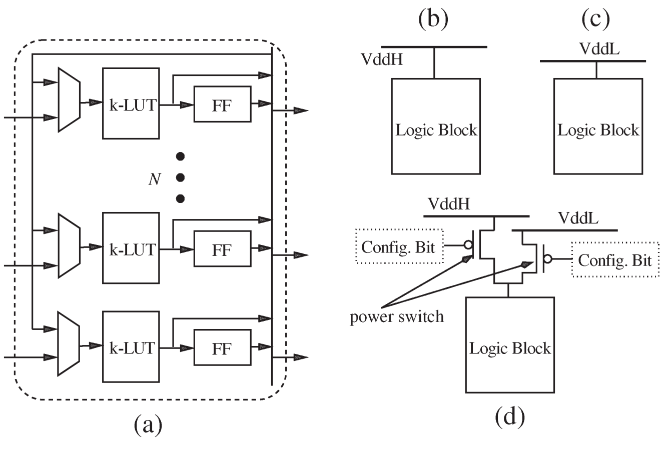


Fig. 3. Logic blocks in dual-Vdd and Vdd-programmable FPGAs. (a) Logic block. (b) H-block. (c) L-block. (d) P-block.

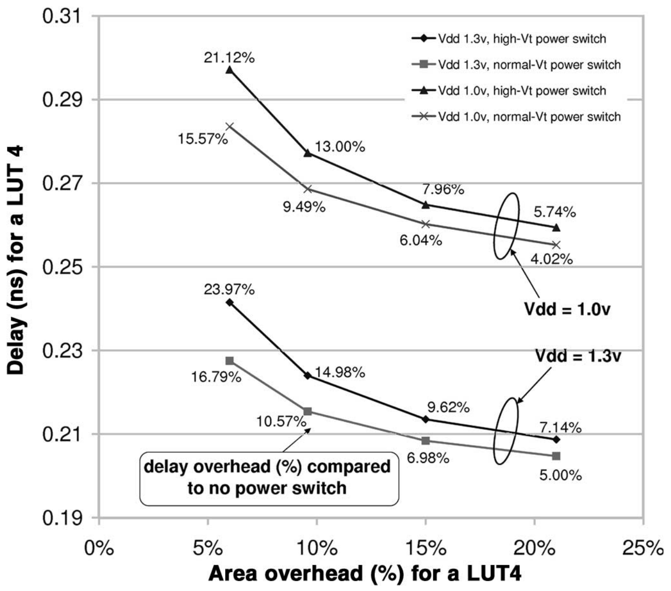


Fig. 4. Area and delay overhead of the power switch for a four-input LUT.

low-leakage SRAM cells can also be used for the programmability of interconnects. In the rest of this paper, we assume that the low-leakage SRAMs are used for all the programmability in FPGAs.

III. Vdd-Programmable Logic Fabrics

To introduce dual-Vdd and Vdd programmability to the logic fabric, we design the detailed circuits and architectures. We assume that the global interconnects use uniform VddH in this section but remove this assumption in Section VI.

A. *Circuit Design for Logic Blocks*

We design three types of logic blocks, as shown in Fig. 3. The first two types "H-block" and "L-block" are connected to supply voltages VddH and VddL, respectively. An H-block has the highest performance, but an L-block has reduced power consumption at the cost of increased delay. To further introduce Vdd programmability, a "P-block" in Fig. 3(d) is designed by inserting p-channel MOS (PMOS) transistors between the power supply rails and the logic block. These transistors are named "power switches," and configuration bits are used to control the power switches so that an appropriate supply voltage can be chosen for the P-block.

Note that the power switch is very similar to the sleep transistor used in power gating [21]. An important design aspect for the sleep transistor is how to determine its size because it impacts both performance and area overhead. We control the area overhead of power switches in two ways. First, sleep transistors usually use high Vt for better leakage reduction in power-off state. Transistors with high Vt have larger ON-resistance and the transistor size needs to be increased for a specified performance. We design power switches with normal Vt so that area overhead can be reduced. Fig. 4 presents the simulation program with integrated circuit emphasis (SPICE) simulation results for a four-input LUT with a power switch.

We use the Berkley Predictive BSIM4 model [22] in 100-nm technology, which considers deep submicrometer effects such as gate leakage. The x axis is the power-switch area in the percentage of the original four-input LUT area, and y axis is the corresponding circuit delay. The area is calculated as the equivalent number of minimum-width transistors. The delay overhead due to power-switch insertion is labeled beside each data point. Under the same area budget, a power switch with normal Vt has smaller delay compared to a power switch with high Vt. Simulation result shows that compared to a normal four-input LUT at the same Vdd level, an optimized four-input LUT with power switches has 5% extra delay and 21% transistor-area overhead.

Because the peak current for a circuit is normally smaller than the sum of peak current for all of its subcircuits, we assume that a logic block shares one power switch, and we carry out SPICE simulations to find the worst-case peak current for the logic block. Most of the LUTs in a logic block are switching in this worst case. We then decide the power-switch size subject to the delay constraint under the worst-case peak current and apply the size to the power switches for all logic blocks. The same principle is applied to power-switch sizing in [21], and novel methods such as the genetic algorithm and the Automatic Test Pattern Generation algorithm [23] can be used to find the worst-case peak current for a logic block. For our logic block with ten LUTs, only 12% area overhead is required to afford the same 5% performance loss for the worst-case simultaneous switching current. Therefore, large granularity significantly reduces the power-switch size and transistor-area overhead. To select different supply voltages, we need two power switches for each logic block. According to Fig. 4, different supply voltages have little impact on the area overhead under the same delay increase. To limit the delay increase to 5%, a P-block logic block requires 24% area overhead for two power switches again for the worst-case simultaneous switching current.

The configuration bits for power switches are stored in SRAM cells and consume leakage power. The dynamic power overhead due to the parasitic capacitances of power switches is almost ignorable as indicated by the SPICE simulation. This is because the power-switch transistor is either ON or OFF during normal operation, and almost no charging or discharging occurs on the source/drain capacitors. Therefore, the major power overhead of Vdd programmability comes from the leakage power of those configuration SRAM cells. Because high-Vt SRAM can be applied to all configuration bits, the increased number of SRAM cells due to supply-voltage programmability may not necessarily lead to a large increase of leakage power.

Due to the similarity between power switches and sleep transistors, we can apply power gating to an unused P-block simply by turning off both switches. However, our normal-Vt power switches may consume more leakage compared to high-Vt sleep transistors. To reduce leakage power effectively, we propose "gate-boosted power switches." When putting a P-block into power-off state, we drive the gate voltage of a PMOS power switch to one Vt higher than the Vdd level at its source node. Table I shows that a gate-boosted power switch can reduce leakage by two orders of magnitude compared to a normal switch. A Vdd-programmable logic block with gate-boosted power switch requires three different Vdd levels. Note that gate boosting has already been used in some commercial Xilinx FPGAs [18] to compensate the logic 1 degradation of n-channel MOS (NMOS) pass transistors in routing switches. Therefore, our study assumes that gate boosting is available for PMOS power switches.

TABLE I
LEAKAGE POWER FOR A P-BLOCK CONTAINING ONE FOUR-INPUT LUT

Vdd	Leakage power (watt)		
	Power-on state	Power-off state	
		normal power switch	gate-boosted power switch
1.3v	2.47E-06	3.46E-07	2.17E-09
1.0v	8.05E-07	3.37E-07	9.28E-10

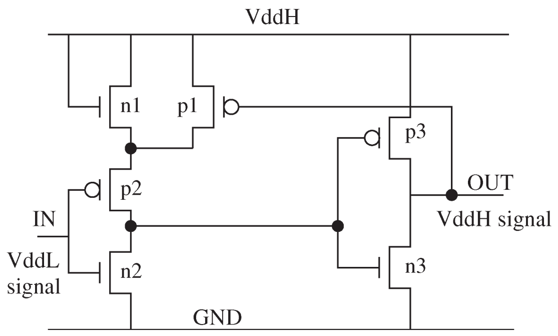


Fig. 5. Level-converter circuit with single supply voltage.

TABLE II
DELAY AND POWER OF THE LEVEL CONVERTER IN FIG. 5 AT BERKLEY-PREDICTED 100-nm TECHNOLOGY

VddH/VddL	delay (ns)	energy per switch (fJ)	leakage power (uW)
1.3v/1.0v	0.0814	7.40	0.0104
1.3v/0.9v	0.0801	8.05	0.0139
1.3v/0.8v	0.0845	9.73	0.0240

B. Level Converter

In a dual-Vdd circuit, the interface between a VddL device and a VddH device must be designed carefully to avoid the excessive leakage power. If a VddL device drives a VddH device and the VddL device output is logic 1, both PMOS and NMOS transistors in the VddH device will be partially ON and will dissipate an unacceptable amount of leakage power due to dc short circuit current. A level converter should be inserted to block the short-circuit current. The level converter converts a VddL signal swing to a VddH signal swing.[2] Different level-converter circuits have been used in dual-Vdd ASIC designs [24]–[27]. We use the recently proposed asynchronous level converters with single supply voltage [28] in our dual-Vdd fabrics. Fig. 5 shows the transistor-level schematic of the level converter. When the input signal is logic 1, the threshold-voltage drop across NMOS transistor n1 can provide a virtual low supply voltage to the first-stage inverter (p2, n2) so that p2 and n2 are not be partially ON. When the input signal is logic 0, the feedback path from node OUT to PMOS transistor p1 pulls up the virtual supply voltage to VddH, and inverter

[2]Note that a VddH device can drive a VddL device without generating excessive leakage power, and no level converter is needed.

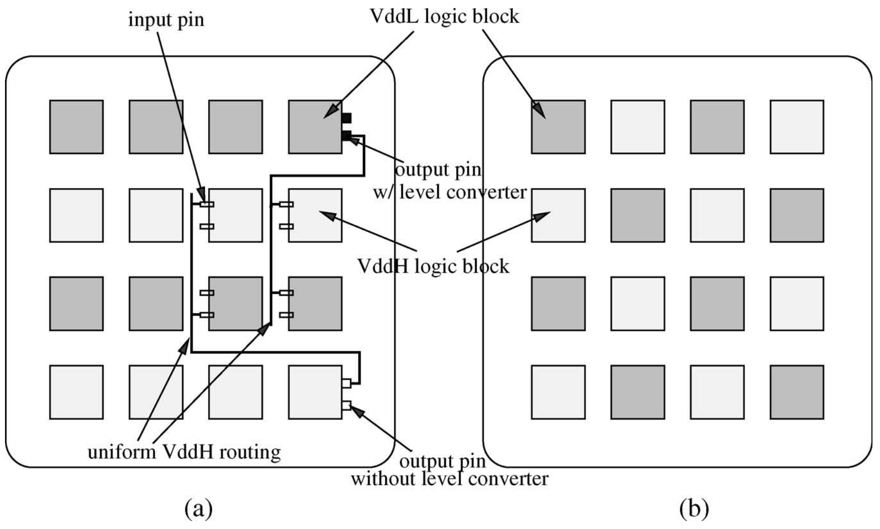


Fig. 6. Predefined dual-Vdd layout patterns. (a) Row-based dual-Vdd layout pattern. (Ratio VddL Row/VddH Row = 1 : 1). (b) Interleaved dual Vdd layout pattern (Ratio VddL Block/VddH Block = 1 : 1).

(p2, n2) generates a VddH signal to the second inverter so that no dc short circuit current occurs. For a particular VddH/VddL combination, we decide the transistor size in the level converter as follows: We start from a level converter with minimum transistor sizes. We then size up the transistors to limit the level-converter delay within 30% of a single LUT delay or 7% of a logic cluster delay. For transistor sizes that meet the delay bound, we choose the sizing with the lowest power consumption. Table II shows the delay and leakage power of the sized level converters. Note that the leakage power increases as the voltage difference between VddH and VddL increases. This is because the threshold-voltage drop cannot provide proper low supply when the gap between VddH and VddL is large. Therefore, the VddH/VddL ratio cannot be too large unless the threshold voltage of NMOS transistor n1 can be adjusted for a given range of VddH/VddL ratio.

C. *Dual-Vdd Logic Fabric and Vdd Programmability*

The combination of three types of logic blocks can construct different FPGA logic fabrics. We study two logic fabrics, a predefined dual-Vdd fabric, and a Vdd-programmable fabric in the succeeding subsections.

1) Predefined Dual-Vdd Fabric: The predefined dual-Vdd fabric named "arch-DV" in this paper is a mixture of H-blocks and L-blocks. There is a predefined Vdd level (VddH or VddL) for each logic block in the fabric. The physical locations of H-blocks and L-blocks define the dual-Vdd layout pattern. Fig. 6 shows two possible layout patterns. One is the row-based pattern with a ratio of VddL-row/VddH-row as 1 : 1. Another is the interleaved layout pattern with a ratio of VddL-block/VddH-block as 1 : 1. The ratio of VddL-row/VddH-row or the ratio of VddL-block/VddH-block can be determined experimentally. Note that the routing resources use uniform VddH because this section focuses on applying dual Vdd only to logic blocks, and dual-Vdd routing fabric is explored in Section VI. Fig. 6 also shows routing paths that connect logic blocks with different supply voltages. The output signals from a VddL logic block must go through level converters before entering the routing channels. If the VddL logic block size is N, i.e., it has N output pins, we need N level converters at the output pins. On the other hand, VddH logic blocks do not need any level converters. The signal in the uniform VddH routing finally reaches another logic block, which can be VddH or VddL. In either case, no level converters are needed at the input pins of a logic block.

2) Vdd-Programmable Logic Fabric: When all the logic blocks in an FPGA fabric are P-blocks, we call it a logic fabric with full Vdd programmability. This logic fabric has the maximum Vdd programmability for logic blocks. For each output of a P-block, we have a level converter to implement the interface from VddL logic block to VddH routing channels. The logic block output can be programmed to either go through the level-converter circuit or bypass it.

On the other hand, one can also obtain a logic fabric by mixing all three types of logic blocks, and it has the level of Vdd programmability between the predefined dual-Vdd logic fabric and the logic fabric with 100% P-blocks. Although such a fabric represents a tradeoff between power-switch area and Vdd programmability, the different tile size of H-block/L-block and P-block may cause difficulty in obtaining a regular fabric layout. In this paper, we only study Vdd-programmable logic fabric with 100% P-blocks, with the experimental results for fabrics of mixed H-block/L-block/P-block being presented in [9].

IV. CAD Flow for Vdd-Programmable Logic Fabrics

We develop a design flow in Fig. 7 to leverage the dual-Vdd and Vdd-programmable fabrics. Given a single-Vdd gate-level netlist, we first apply single-Vdd technology mapping and timing-driven packing [18] to obtain a cluster-level netlist. We then perform single-Vdd timing-driven placement and routing by the Versatile Place and Route Tool for FPGAs (VPR) [18] and generate the back-annotated basic circuit netlist (BC-netlist) defined in [2]. As the first step to consider dual Vdd in the design flow, Vdd assignment for logic blocks is performed

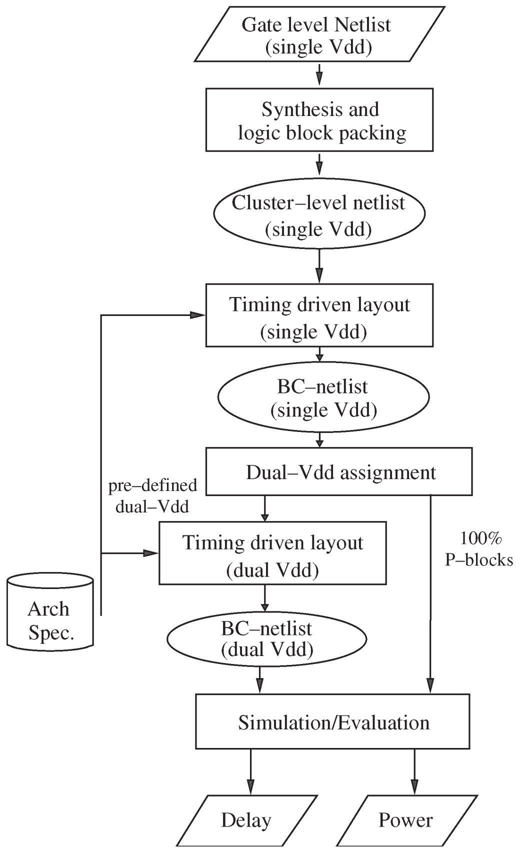

Gate level Netlist (single Vdd)
Synthesis and logic block packing
Cluster–level netlist (single Vdd)
Timing driven layout (single Vdd)
BC–netlist (single Vdd)
Dual–Vdd assignment
pre–defined dual–Vdd
Timing driven layout (dual Vdd)
100% P–blocks
Arch Spec.
BC–netlist (dual Vdd)
Simulation/Evaluation
Delay
Power

Fig. 7. Design flow for predefined dual-Vdd and Vdd-programmable logic fabrics.

to obtain a dual-Vdd BC-netlist. After the dual-Vdd assignment, we have two different design paths. If the logic fabric has full Vdd programmability (i.e., 100% P-blocks), the dual-Vdd assignment result is always feasible, and an enhanced version of the FPGA power analysis framework “fpgaEva-LP” [2], [14] is used to estimate the power and performance. If the logic fabric is a predefined dual-Vdd fabric, the corresponding design path goes through an extra step of dual-Vdd placement.[3] We discuss the dual-Vdd assignment and dual-Vdd placement in the succeeding sections.

A. Dual-Vdd Assignment

The dual-Vdd assignment determines the Vdd level for each logic block in the mapped netlist. It makes use of the time slack in a circuit and performs power optimization by applying dual-Vdd levels. Sensitivity-based algorithms have been used in ASIC circuit tuning for either delay optimization [29] or power-delay tradeoff [30]. We use a similar sensitivity-based algorithm for dual-Vdd assignment. First, we define “power sensitivity” as follows.

Definition 1 (Power Sensitivity S_x): For a given design variable x, power sensitivity is calculated as

$$S_x = \frac{\Delta P}{\Delta x} = \frac{\Delta P_{\text{sw}}}{\Delta x} + \frac{\Delta P_{\text{lkg}}}{\Delta x} \tag{1}$$

[3]Because we apply uniform high Vdd (VddH) to interconnects in this section, the routing algorithm does not need to consider dual Vdd.

```
Sensitivity-based dual-Vdd assignment algorithm:
input: single-Vdd BC-netlist $N$
output: dual-Vdd BC-netlist $N'$
          (with original Vdd and another low Vdd)
constraint:
  $\frac{crit\_path\_delay(N')-crit\_path\_delay(N)}{crit\_path\_delay(N)} < delay\_increase\_bound$

Let $N_p$ be the input netlist $N$;
While( $N_p$ has logic blocks not tried )
begin
    Calculate power-sensitivity for logic blocks not tried;
    Select logic block $B$ with largest sensitivity;
    Assign low Vdd to $B$ and update timing information;
    If( delay constraint not met )
        Reverse the low-Vdd assignment;
    mark logic block $B$ as ‘tried’;
end
Let the output netlist $N'$ be $N_p$
```

Fig. 8. Sensitivity-based dual-Vdd assignment algorithm.

where P_{sw} is switching power and P_{lkg} is leakage power.

In our dual-Vdd assignment problem, the design variable x becomes supply voltage Vdd. To calculate power sensitivity, we need the relationship between power and supply voltage. We use the FPGA power model in [2]. The switching power P_{sw} of a primitive node i in the BC-netlist is calculated as follows:

$$P_{\text{sw}}(i) = 0.5f \cdot \hat{E}_i \cdot C_i \cdot \text{Vdd}^2 \tag{2}$$

where f is the clock frequency, $\hat{E}_i$ is the effective transition density considering glitches, and C_i is the load capacitance. The leakage power P_{lkg} of node i is calculated as follows:

$$P_{\text{lkg}}(i) = I_{\text{lkg}}(\text{Vdd}) \cdot \text{Vdd} \tag{3}$$

where I_{lkg} is the leakage current at supply voltage Vdd. The power sensitivity of a logic block B can be calculated as the sum of sensitivities for all the nodes inside this logic block, i.e.,

$$S_x(B) = \sum_{\text{node } i \in B} S_x(i). \tag{4}$$

We present our dual-Vdd assignment algorithm in Fig. 8. It is a greedy algorithm with an iteration loop. Given the single-Vdd BC-netlist, we analyze the timing and obtain the circuit path with the largest time slack. Power sensitivity is calculated for logic blocks on this path but not on the critical path. We assign low Vdd to the logic block with the largest power sensitivity and update the timing information. If the new critical path delay exceeds the user-specified delay increase bound, we reverse the low-Vdd assignment. Otherwise, we keep this assignment and go to the next iteration. In either case, the logic block selected in the current iteration will not be revisited in other iterations. Dual-Vdd assignment ignores the placement constraint imposed by the predefined pattern. This constraint is to be handled by dual-Vdd placement.

B. Placement for Predefined Dual-Vdd Fabric

We develop a dual-Vdd placement algorithm based on the simulated annealing algorithm implemented in VPR [18].

The VPR placement tool models an FPGA as a set of legal slots or discrete locations, at which logic blocks or I/O pads can be placed. A linear congestion cost function is used in VPR placement to reduce wire length and congestion, which is shown as follows:

$$\text{Cost}_{\text{lin-cgst}} = \sum_{i=1}^{N_{\text{nets}}} q(i) \left[\frac{bb_x(i)}{C_{\text{av},x}(i)^{\gamma}} + \frac{bb_y(i)}{C_{\text{av},y}(i)^{\gamma}} \right]. \qquad (5)$$

The summation is over the number of nets N_{nets} in the circuit. For each net i, $bb_x(i)$ and $bb_y(i)$ represent the horizontal and vertical spans of its bounding box, respectively. The $q(i)$ compensating factor is due to the fact that a wire length model using the bounding box underestimates the wiring required to connect nets with more than three terminals. The value of $q(i)$ depends on the number of terminals in net i. $C_{\text{av},x(i)}$ and $C_{\text{av},y(i)}$ are the average channel capacities in the x and y directions, respectively, over the bounding box of net i. When channel capacities are different across an FPGA chip, the cost function penalizes placements, which require more routing in the narrower channels and hence reduce the routing congestion.

We adopt the same adaptive annealing schedule in VPR but use a new cost function that has terms considering dual Vdd. "moves" are defined as either swapping two logic blocks or relocating a logic block to an empty slot. The cost of relocating a logic block j to an empty slot, which is also the cost difference between the placements before and after the relocation,[4] is

$$\begin{aligned}\text{Cost(relocate)} &= \Delta\text{Cost(placement)} \\ &= \Delta\text{Cost}_{\text{lin-cgst}} - \alpha \cdot \Delta\text{matched}(j) \\ &\quad + \beta \cdot (1 - \text{matched}(j)). \qquad (6)\end{aligned}$$

Weight coefficients α and β are determined experimentally. In our experiments, we find that $\alpha = 1 \times 10^{-5}$ and $\beta = 0.5 \times 10^{-5}$ can push most of the logic blocks into Vdd-matched sites while introducing minimal wire length and delay penalty. matched(j) is a Boolean function that describes the Vdd-matching status of a logic block in the new slot and is defined as

$$\text{matched}(j) = \begin{cases} 1, & \text{VddL block } j \text{ in the slot of L-block or P-block} \\ 1, & \text{VddH block } j \text{ in the slot of H-block or P-block} \\ 0, & \text{otherwise.} \end{cases}$$

If the Vdd assigned to block j matches the Vdd at its physical location, matched(j) returns a value of "1." Otherwise, it returns "0." Because the power supply of a P-block in a Vdd-programmable fabric is configurable, any logic block placed in a P-block slot returns a matched value. Δmatched(j) is the difference in matched(j), which penalizes relocating block j from a Vdd-matched location to an unmatched location. The term $1 -$ matched(j) penalizes relocating block j from a Vdd-unmatched location to another unmatched location and attempts to maximize the number of Vdd-matched logic blocks. Considering the power and delay overhead of a P-block used in a Vdd-programmable logic fabric, we further penalize the Vdd-matched location at a P-block slot other than that at an H-block or L-block slot. The cost of swapping two logic blocks is the sum of the costs given by (6) for the two blocks. Because the time overhead to calculate the new cost function is almost ignorable, our dual-Vdd placement algorithm runs in the same time as the original VPR placement tool.

[4]We use this representation of our cost function because it is easier to integrate into the incremental updating mechanism in the simulated annealing.

TABLE III
PERCENTAGE OF VddL LOGIC BLOCKS GIVEN BY DUAL-Vdd WITH ZERO DELAY INCREASE AND NO LAYOUT RESTRICTIONS (VddH = 1.3 V, AND VddL = 0.8 V)

circuit	# of logic blocks	# of I/O blocks	% of VddL logic blocks
alu4	162	22	74.07
apex2	213	41	46.01
apex4	134	28	60.45
bigkey	294	426	89.12
clma	1358	144	80.93
des	218	501	74.31
diffeq	195	103	83.59
dsip	588	426	54.32
elliptic	666	245	90.74
ex1010	513	20	75.66
ex5p	194	71	60.98
frisc	731	136	95.13
misex3	181	28	57.52
pdc	624	56	69.54
s298	266	10	82.81
s38417	982	135	88.67
s38584	1046	342	96.73
seq	274	76	53.03
spla	461	122	79.70
tseng	305	174	86.26
Avg			74.98

V. EXPERIMENTAL RESULTS FOR VDD-PROGRAMMABLE LOGIC FABRICS

In this section, we first compare four types of FPGAs, namely: 1) arch-SV; 2) arch-DV; 3) arch-PV; and 4) ideal-DV. The difference between the four FPGAs lies in the logic fabric. Their interconnect fabrics all use uniform VddH. The logic fabric in arch-SV uses the same single Vdd as its interconnect fabric, and it is the baseline in our architecture comparison. arch-DV is our predefined dual-Vdd FPGA, and its logic fabric consists of H-blocks and L-blocks, with VddH being the same Vdd level for its interconnect fabric. Both row-based and interleaved dual-Vdd layout patterns are studied. arch-PV is our Vdd-programmable FPGA with 100% P-blocks. ideal-DV is the ideal case that does not have any P-blocks but assumes that the mixture and placement of H-blocks and L-blocks can be "perfectly" customized for each individual application. Compared to arch-PV with 100% P-blocks to customize the power supply for every logic block, ideal-DV has neither power and delay overhead associated with P-blocks nor the capability to turn off the unused logic blocks by power gating. It does have all the necessary level converters whenever a VddL block drives a VddH block.

Before we present the experimental results, we need to determine the ratio between the H-block and L-block for predefined dual-Vdd FPGA arch-DV. Table III shows the percentage of VddL logic blocks after dual-Vdd assignment for 20 MCNC benchmark circuits [31]. No delay increase is allowed when we

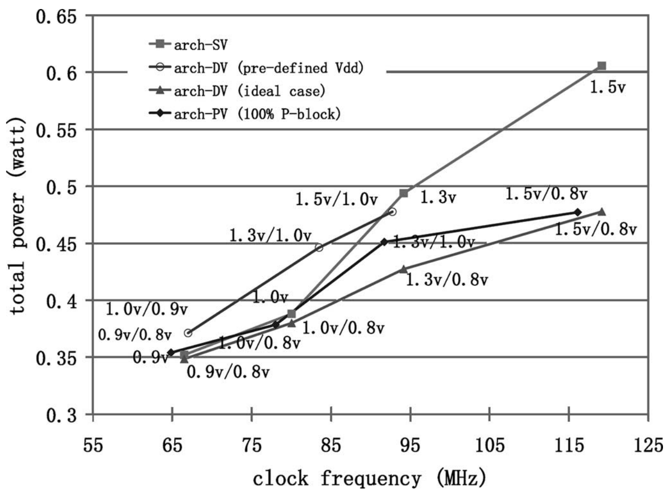


Fig. 9. Power versus delay for s38584 (arch-SV, single-Vdd FPGA; arch-DV, FPGA with predefined dual-Vdd logic fabric; arch-PV, FPGA with Vdd-programmable logic fabric).

assign VddL to logic blocks. VddH and VddL are set to 1.3 and 0.8 V, respectively. The percentage varies from 53% to 96%, and the average is about 75%. It clearly shows that circuits implemented on uniform-Vdd FPGA have a large amount of timing slack to be utilized for power reduction. According to the ideal percentage given by dual-Vdd assignment and considering that the predefined dual-Vdd layout pattern constraints may reduce the percentage of VddL logic block, we set the ratio H : L to 1 : 2 for arch-DV.[5]

A. Architecture Comparison

We carry out experiments on 20 MCNC benchmarks for the four types of FPGAs. Both row-based and interleaved layout patterns in Fig. 6 have been tried for arch-DV. However, our experimental results show no significant power and performance difference between these two layout patterns. Considering that a row-based layout pattern is easier to use in routing the power/ground network, we only present the experimental results of the row-based layout pattern for arch-DV.

Fig. 9 presents the architecture comparison for a large MCNC benchmark circuit "s38584." The x axis is the clock frequency calculated as the reciprocal of critical path delay. This delay is obtained by the timing analysis in VPR based on the Elmore delay model and is generally overestimated. The y axis is the total power consumption. Each curve in the figures represents the power-versus-performance tradeoff for a particular FPGA architecture under different Vdd levels or VddH/VddL combinations. For arch-DV and ideal-DV, we try several different VddH/VddL combinations and prune the "inferior" data points (i.e., those with larger power consumption and smaller clock frequency compared to certain VddH/VddL combination) to obtain the curve. We label the Vdd level or VddH/VddL combination beside each data point. The curve for arch-SV shows that we can scale down the Vdd level of a single-Vdd FPGA and reduce the power at the cost of performance degradation.[6] The curve for arch-DV demonstrates poor performance for the predefined dual-Vdd FPGA. The placement constraint for the predefined dual-Vdd fabric arch-DV is large enough to degrade the performance dramatically and equivalently leads to a large power overhead at the same clock frequency.

With Vdd programmability to remove the placement constraint of matching Vdd levels, FPGA arch-PV (with 100% P-blocks) is able to achieve a better power-versus-performance tradeoff curve compared to arch-SV (see Fig. 9). The advantage of FPGA architecture arch-PV over arch-DV has been observed for all the benchmark circuits, and it shows that the field programmability of Vdd is required to obtain a satisfactory power-versus-performance tradeoff. In the clock frequency range of our experiments, the power saving by arch-PV is larger at the higher frequency end. This is because higher clock frequency usually requires higher supply voltage, and the gap between VddH and VddL could be larger. As VddH decreases at lower frequency and VddL is limited by the lowest Vt in a technology, the gap between VddH and VddL decreases as well. This limits the opportunity for dual-Vdd optimization at the lower frequency end. Moreover, FPGA arch-PV gives a power-versus-performance tradeoff curve that is close to that of FPGA ideal-DV for most VddH/VddL combinations. It shows that the power and delay overhead for Vdd programmability is relatively small.

Table IV presents the power saving by Vdd-programmable FPGA arch-PV as well as the delay increase when compared to single-Vdd FPGA arch-SV for all the MCNC benchmark circuits [32]. The Vdd level for arch-SV is 1.3 V, as suggested by the International Technology Roadmap for Semiconductors (ITRS) [33] for 100-nm technology. The VddH/VddL combination for arch-PV is 1.3 V/0.8 V. FPGA arch-PV has a larger critical path delay due to the insertion of power switch for logic blocks. However, this delay increase is very small with properly sized power switches, and it is only 2.33% in our experiments. We break down FPGA power into logic power, local interconnect power, and global interconnect power. The logic power is the power of LUTs, flip-flops, and MUXs in logic blocks. The local interconnect power is the power of internal routing wires and buffers within logic blocks. Routing wires outside logic blocks, programmable interconnect switches in routing channels, and their configuration SRAM cells contribute to the global interconnect power. Because FPGA arch-PV has the Vdd programmability for logic blocks, it can reduce both logic power and local interconnect power. On average, arch-PV reduces logic power by 42.30% and reduces local interconnect power by 37.97%. However, the total FPGA power saving is significantly smaller, and it is only 9.74% on average. When considering the delay increase in FPGA arch-PV, the energy-delay product reduction is only 5.48%. The small power saving for an entire FPGA chip is because global interconnects between logic blocks consume most of the power in an FPGA.

[5]We also conducted experiments with higher percentage of H-blocks. The experimental results do not change the conclusion to be presented that a fixed dual-Vdd layout pattern leads to a large delay penalty.

[6]For Vdd scaling, we assume that the threshold voltage Vt can be scaled to maintain constant leakage. This scheme is called constant leakage scaling, and it offers a good power and performance tradeoff, as studied in [8].

TABLE IV
POWER AND DELAY COMPARISON BETWEEN FPGA arch-PV (WITH 100% P-BLOCKS) AND THE BASELINE FPGA arch-SV. THE Vdd IS 1.3 V FOR arch-SV, AND THE VddH/VddL COMBINATION IS 1.3 V/0.8 V FOR arch-PV

circuit	arch-SV (baseline)				arch-PV (100% P-blocks)				
	delay (ns)	logic power (Watt)	local intcnt. power (Watt)	global intcnt power (Watt)	delay increase	logic power saving	local intct. power saving	total FPGA power saving	FPGA energy-delay product reduction
alu4	10.38	0.0175	0.0300	0.0974	2.13%	43.90%	42.34%	15.31%	11.66%
apex2	11.10	0.0192	0.0321	0.1528	1.99%	27.95%	27.24%	7.68%	3.98%
apex4	10.13	0.0101	0.0147	0.0807	2.12%	37.73%	36.39%	9.33%	5.45%
bigkey	6.34	0.0603	0.0788	0.2267	1.97%	60.10%	47.13%	21.31%	18.19%
clma	21.37	0.0474	0.0550	0.9147	2.33%	35.33%	33.34%	2.11%	-2.52%
des	10.60	0.0526	0.0542	0.3183	2.20%	63.85%	40.48%	14.37%	10.55%
diffeq	11.93	0.0076	0.0071	0.0580	3.49%	36.99%	39.03%	8.27%	1.76%
dsip	5.49	0.0434	0.0564	0.2635	2.14%	53.53%	29.80%	12.26%	8.46%
elliptic	16.02	0.0176	0.0174	0.2046	2.81%	43.44%	41.34%	7.18%	1.90%
ex1010	14.69	0.0229	0.0266	0.2818	1.98%	39.08%	37.98%	6.27%	2.52%
ex5p	11.49	0.0084	0.0117	0.0772	2.10%	37.33%	33.07%	8.23%	4.34%
frisc	22.30	0.0185	0.0141	0.3188	2.72%	36.74%	33.08%	3.54%	-1.79%
misex3	9.65	0.0158	0.0264	0.1092	2.21%	36.72%	35.43%	11.57%	7.61%
pdc	14.67	0.0266	0.0359	0.4234	1.80%	34.96%	36.96%	5.22%	1.78%
s298	21.16	0.0104	0.0152	0.0813	2.30%	40.28%	43.41%	12.19%	8.11%
s38417	14.63	0.0423	0.0527	0.3842	2.71%	43.40%	37.27%	7.23%	2.14%
s38584	10.62	0.0484	0.0739	0.3700	2.68%	54.06%	51.41%	15.57%	10.98%
seq	9.31	0.0204	0.0335	0.1521	2.13%	34.18%	32.88%	10.07%	6.20%
spla	13.77	0.0210	0.0273	0.2606	1.80%	41.48%	43.26%	7.45%	4.10%
tseng	12.47	0.0073	0.0082	0.0450	2.94%	44.88%	37.55%	9.61%	4.22%
Avg.	-	-	-	-	2.33%	42.30%	37.97%	9.74%	5.48%

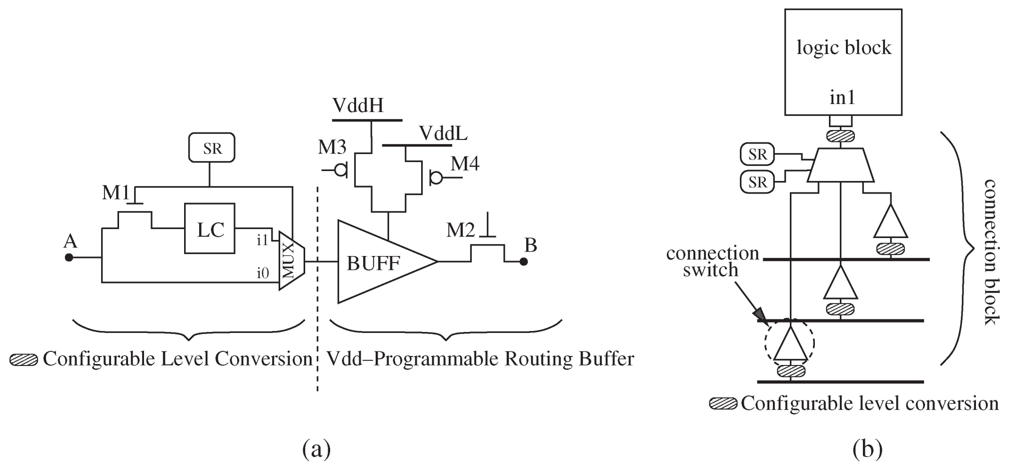

logic block
in1
SR
SR
connection block
connection switch
▨ Configurable level conversion
VddH
VddL
M3
M4
SR
M1
LC
i1
i0
MUX
M2
A
B
BUFF
▨ Configurable Level Conversion
Vdd–Programmable Routing Buffer
(a)
(b)

Fig. 10. (a) Vdd-programmable routing switch. (b) MUX-based Vdd-programmable connection block. ["SR" stands for SRAM cell, and "LC" stands for the level converter. The same configurable level-conversion circuit is used in both (a) and (b)].

As shown by the power breakdown in Table IV for arch-SV, the global interconnect power is significantly larger than the sum of logic power and local interconnect power. Therefore, Vdd programmability must be applied to FPGA interconnect fabric in order to achieve significant total power saving, which is to be presented in the next section.

VI. VDD-PROGRAMMABLE INTERCONNECT FABRICS

A. *Vdd-Programmable Interconnect Fabric*

We apply programmable dual-Vdd to each interconnect switch (either a routing switch or a connection switch). Our Vdd-programmable routing switch is shown in Fig. 10(a). The right part of the circuit is the Vdd-programmable routing buffer. For the tristate buffer in the routing switch, we insert two PMOS transistors M3 and M4 between the tristate buffer and the VddH and VddL power rails, respectively. Similar to a Vdd-programmable logic block, turning off one of the two power

TABLE V
UTILIZATION RATE OF INTERCONNECT SWITCHES

circuit	total interconnect switches	unused interconnect switches	utilization rate
alu4	36478	31224	14.40%
apex4	43741	37703	13.80%
bigkey	63259	57017	9.87%
clma	653181	593343	9.16%
des	87877	79932	9.04%
diffeq	42746	36974	13.50%
dsip	75547	70138	7.16%
elliptic	140296	125800	10.33%
ex5p	45404	39288	13.47%
frisc	238853	216993	9.15%
misex3	39928	33819	15.30%
pdc	268167	238610	11.02%
s298	43725	37641	13.91%
s38417	243315	216577	10.99%
s38584	195363	174460	10.70%
seq	61344	53173	13.32%
spla	153235	134991	11.91%
tseng	29051	25026	13.85%
Avg.			11.90%

TABLE VI
DELAY AND POWER OF A Vdd-PROGRAMMABLE ROUTING SWITCH. WE USE A 7× MINIMUM-WIDTH TRISTATE BUFFER FOR ROUTING SWITCHES AND A 4× MINIMUM-WIDTH PMOS TRANSISTOR FOR POWER SWITCHES

Routing Buffer				
Vdd	delay (sec)		energy per switch (Joule)	
	without power switches	with power switches (increase %)	without power switches	with power switches
1.3v	5.90E-11	6.86E-11 (+16.27%)	3.3049E-14	3.2501E-14
1.0v	6.45E-11	7.55E-11 (+17.05%)	1.6320E-14	1.6589E-14

Delay of the Level Converter preceding a Routing Buffer		
configuration	Configurable Level Converter	Normal Level Converter in Fig. 5
1.0v → 1.3v (level conversion)	1.0613E-10	0.814E-10
1.3v → 1.3v (conversion by pass)	9.0060E-12	N/A

switches can select a Vdd level for the routing switch. Considering the extremely low interconnect utilization rate (on average, 11.90%,[7] as shown in Table V, for the MCNC benchmark set), we can turn off both power switches and power gate an unused routing switch. In that case, we provide three Vdd states, namely: 1) high Vdd; 2) low Vdd; and 3) power gating.

The power-gating state provided by Vdd programmability is very attractive because our SPICE simulation shows that the power gating of the routing switch can reduce its leakage power by a factor of over 300 at circuit level. We also consider the power and delay overhead associated with the power-switch insertion. The dynamic power overhead is almost ignorable (see the energy per switch in Table VI). This is because the power switches stay either ON or OFF, and there is no charging and discharging at their source/drain capacitors. The main power overhead is the leakage power of the extra configuration cells for Vdd selection. We use the same high-Vt SRAM cells in Section II to reduce configuration cell leakage. Furthermore, the Vdd-programmable routing buffer has an increased delay compared to the conventional routing buffer because the power switches are inserted between the buffer and power supply. We size the power switches for the tristate buffer to achieve a bounded delay increase. For a routing architecture with all wire segments spanning four logic blocks, we assume 7× minimum-width tristate buffers and get a 16% delay increase by inserting 4× minimum-width power switches. The left part of the circuit in Fig. 10(a) is the configurable level converter. We insert the level converter right before the routing buffer and use a MUX to either use this level converter or bypass it. Transistor M1 is used to prevent signal transitions from propagating through the level converter when it is bypassed; therefore, the dynamic power of an unused level converter is eliminated. Only one configuration bit is needed to realize the level-converter selection and signal gating for unused level converters. The delay of this configurable level converter is also shown in Table VI, in contrast with a normal level converter.

Connection block is another type of routing resource [18]. Fig. 10(b) shows the MUX-based implementation of a connection block, which chooses only one track in the channel and connects it to the logic-block input pin. The buffers between the routing track and the MUX are connection switches. To apply Vdd programmability to connection blocks, we can simply replace the connection switches with Vdd-programmable buffers and insert the configurable level conversion circuits before each new connection switch. However, this structure introduces a large number of extra configuration SRAM cells. For a connection block containing N Vdd-programmable connection switches, there are $2N + \lceil \log_2 N \rceil$ configuration SRAM cells, among which the $\lceil \log_2 N \rceil$ SRAM cells are for the MUX and the other $2N$ SRAM cells are for the N Vdd-programmable connection switches. Another disadvantage for such a connection block is that its delay increases quickly as the number of inputs to the connection block increases. Recently, new Vdd-programmable switch and connection block are proposed in [14] to improve the SRAM efficiency as well as the delay. As shown in Fig. 11(a), a Vdd-programmable switch module with three signal ports, namely: 1) $VddH\_En$; 2) $VddL\_En$; and 3) $Pass\_En$, is first defined. By setting these three control signals, we can program the Vdd-programmable switch between Vdd selection and power gating. Fig. 11(b) further shows the SRAM-efficient Vdd-programmable switch. $Pass\_En$ can be generated by $VddH\_En$ and $VddL\_En$ with a NAND2 gate. With the new Vdd-programmable switch, the SRAM-efficient design of Vdd-programmable connection block is shown in Fig. 11(c). It removes the MUX and tie the tristate outputs of Vdd-programmable switches together. A $\lceil \log_2 N \rceil : N$ decoder and $2N$ NAND2 gates are used to generate the control signals for the Vdd-programmable switches. When the connection block is used, only one of the Vdd-programmable switches is enabled, and the Vdd_sel signal selects its Vdd level. The other Vdd-programmable switches are power gated. When the entire connection block is not used, $Dec\_Disable$ is asserted to power gate all the Vdd-programmable switches in the connection block. By combining the configuration bits for signal multiplexing in a MUX and those for Vdd selection, this design avoids a straightforward implementation of Vdd-programmability on connection blocks and reduces the number of SRAM cells for an N-input connection block to $\lceil \log 2N \rceil + 2$. Due to the removal of pass transistor trees from the critical path, the new connection block is 28% faster and consumes 19% less dynamic power as shown in Table VII. We use the SRAM-efficient Vdd-programmable switches and connection blocks in our interconnect fabric. Because we apply programmable Vdd to both logic blocks and programmable interconnect switches, it is possible that a VddL connection switch connects to a VddH logic block. To ensure that there is supply level conversion for

[7]Note that we use the minimum FPGA array that just fits the application circuit. In reality, the chip size can be significantly larger than necessary, and the interconnect switch utilization can be much lower.

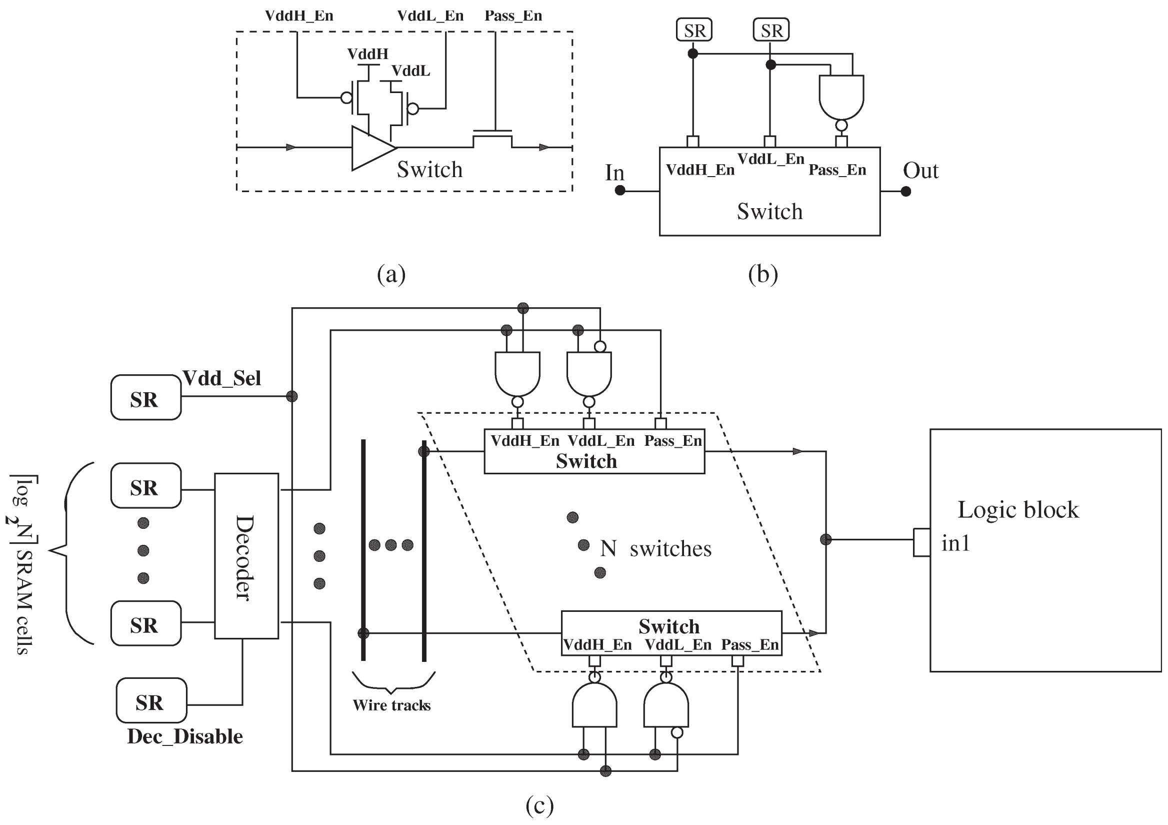


Fig. 11. (a) Vdd-programmable switch. (b) SRAM-efficient Vdd-programmable switch. (c) SRAM-efficient Vdd-programmable connection block.

TABLE VII
DELAY AND POWER OF THE SRAM-EFFICIENT Vdd-PROGRAMMABLE CONNECTION BLOCK COMPARED TO A CONVENTIONAL CONNECTION BLOCK WITHOUT Vdd PROGRAMMABILITY. WE USE A 4× MINIMUM-WIDTH TRISTATE BUFFER FOR CONNECTION SWITCHES AND A 1× MINIMUM-WIDTH PMOS TRANSISTOR FOR POWER TRANSISTORS. THE PARAMETER N FOR A CONNECTION BLOCK IS 32

Vdd	connection block delay (ns)		energy per switch (Joule)	
	w/o Vdd programmability	w/ Vdd programmability (increase %)	w/o Vdd programmability	w/ Vdd programmability (increase %)
1.3v	2.93E-10	2.10E-10 (-28.33%)	3.84E-14	3.11E-14 (-19.01%)
1.0v	3.70E-10	2.22E-10 (-40.00%)	3.09E-14	2.04E-14 (-33.98%)

this type of connection, we also insert the configurable level conversion circuit before each logic-block input pin.

The resulting FPGA has Vdd programmability for both logic and interconnect fabrics, and we name it "arch-PV-fpga." The same design flow for the FPGA arch-PV (with 100% P-blocks) in Fig. 7 can be applied to arch-PV-fpga. The only difference is that a circuit element in dual-Vdd assignment can be either a logic block or an interconnect switch. Power sensitivity is calculated for both logic blocks and interconnect switches. The Vdd assignment is performed as a postrouting procedure, and no change is made to the original VPR tool.

B. Experimental Results

In this section, we compare the new fabric arch-PV-fpga with the baseline fabric arch-SV and present the results in Table VIII. The Vdd-programmable interconnects in FPGA arch-PV-fpga enable Vdd selection for used interconnect switches and power gating for unused interconnect switches. Because the FPGA area is dominated by the interconnect fabric, our fine-grained Vdd-programmable interconnect fabric increases the FPGA tile size significantly. Using the area model in [18], we have estimated that FPGA arch-PV-fpga is 125% larger than baseline FPGA arch-SV [34]. The larger tile size translates into a 50% wire length increase (also a 50% wire capacitance and resistance increase) for each wire segment in routing channels.

As shown in columns 6–8, the leakage power of global interconnects is reduced by 55.51%, and the dynamic power of global interconnects is reduced by 32.39%. The overall global interconnect power is reduced by 46.74%. With this power reduction for global interconnects, arch-PV-fpga is able to reduce the total FPGA power by 47.61%. In contrast, arch-PV only applies Vdd programmability to the logic fabric, and the total FPGA power reduction is only 9.74% (see Table IV). Our arch-PV-fpga has a 17.65% delay increase compared to arch-SV. The majority of delay increase comes from longer wire segments due to the area overhead. The rest delay increase comes from Vdd-programmable routing switches. Considering this performance loss, we compare the metric of the

TABLE VIII
POWER AND DELAY COMPARISON BETWEEN FPGA arch-PV-fpga AND BASELINE FPGA arch-SV. THE Vdd IS 1.3 V FOR arch-SV, AND THE VddH/VddL COMBINATION IS 1.3 V/0.8 V FOR arch-PV-fpga

1	2	3	4	5	6	7	8	9	10	11	12
	arch-SV (baseline)			arch-PV-fpga w/ SRAM efficient connection switches						arch-PV-fpga w/ conventional tree-based connection switches	
circuit	delay (ns)	global intcnt. power (Watt)	total power (Watt)	delay increase	global intcnt. power saving			total FPGA power saving	FPGA ED product reduction	delay increase	ED product reduction
					overall	leakage	dynamic				
alu4	10.38	0.0974	0.1450	18.07%	37.94%	55.68%	23.39%	42.40%	19.71%	24.00%	11.43%
apex2	11.10	0.1528	0.2040	10.19%	36.75%	55.43%	18.68%	37.76%	24.43%	16.09%	16.13%
apex4	10.13	0.0807	0.1056	30.75%	48.42%	55.35%	37.47%	51.15%	16.48%	36.59%	8.85%
bigkey	6.34	0.2267	0.3658	14.79%	40.72%	55.57%	27.90%	46.22%	29.13%	19.64%	23.01%
clma	21.37	0.9147	1.0172	22.52%	52.39%	55.69%	38.36%	51.05%	26.53%	26.63%	21.51%
des	10.60	0.3183	0.4251	12.18%	41.79%	55.79%	27.91%	45.86%	31.87%	17.55%	25.19%
diffeq	11.93	0.0580	0.0726	9.56%	51.95%	54.94%	36.50%	48.82%	38.57%	15.24%	32.04%
dsip	5.49	0.2635	0.3634	14.59%	40.35%	55.89%	25.41%	43.27%	25.51%	20.15%	18.10%
elliptic	16.02	0.2046	0.2395	13.20%	53.31%	55.37%	44.07%	51.30%	37.60%	17.92%	32.29%
ex1010	14.69	0.2818	0.3312	30.60%	50.96%	55.53%	33.67%	50.62%	15.77%	34.63%	10.49%
ex5p	11.49	0.0772	0.0973	33.95%	48.03%	55.42%	34.12%	50.04%	10.36%	40.42%	1.5%
frisc	22.30	0.3188	0.3515	5.05%	54.58%	55.95%	34.48%	52.52%	47.60%	12.33%	40.09%
misex3	9.65	0.1092	0.1514	17.78%	36.88%	55.50%	21.09%	40.78%	17.84%	24.44%	8.29%
pdc	14.67	0.4234	0.4859	21.10%	48.13%	55.75%	21.52%	47.32%	22.74%	25.79%	16.65%
s298	21.16	0.0813	0.1069	16.66%	51.41%	55.93%	40.77%	49.51%	31.28%	21.57%	25.38%
s38417	14.63	0.3842	0.4791	20.30%	49.03%	55.24%	34.77%	49.22%	26.50%	24.49%	21.30%
s38584	10.62	0.3700	0.4923	12.85%	50.45%	55.15%	43.03%	51.09%	37.71%	19.29%	30.40%
seq	9.31	0.1521	0.2061	16.36%	39.81%	55.57%	24.72%	42.70%	22.43%	21.41%	15.54%
spla	13.77	0.2606	0.3089	23.20%	50.12%	55.68%	35.71%	49.98%	24.08%	29.22%	16.48%
tseng	12.47	0.0450	0.0605	9.35%	51.83%	54.86%	44.31%	50.69%	41.03%	15.97%	33.68%
Avg.	-	-	-	17.65%	46.74%	55.51%	32.39%	47.61%	27.36%	23.17%	20.42%

energy-delay product in column 10 and show that arch-PV-fpga can still reduce the energy-delay product significantly (27.36%, on average, for the MCNC benchmark circuits). Note that our SRAM-efficient connection block is 28% faster compared to the MUX-based connection block, and it helps mitigate the delay overhead. Columns 11 and 12 show the delay increase and energy-delay product reduction of arch-PV-fpga when using straightforward implementation of Vdd-programmable connection block based on a tree-structure MUX. At circuit level, the tree-based connection block with Vdd programmability has a 6% larger delay than the baseline connection block without Vdd-programmability. Together with the delay overhead from longer wire segments and Vdd-programmable routing switches, tree-based connection blocks would result in a 23.17% delay increase and only 20.42% energy-delay product reduction. This comparison clearly shows the performance advantage of SRAM-efficient connection blocks in our Vdd-programmable FPGAs.

VII. CONCLUSION AND DISCUSSION

This paper presents field-programmable dual-Vdd and Vdd gating (in short, Vdd programmability) for FPGA power reduction. We have shown that field programmability is required to achieve a satisfactory power-versus-performance tradeoff. We have designed novel FPGA logic and interconnect fabrics with Vdd programmability and developed a simple yet practical CAD flow to leverage the new fabrics. We have carried out a highly quantitative study using delay and power models obtained from circuit design at 100-nm technology. We also estimated that the area overhead of our Vdd-programmable FPGA is 125%, using the area model [18]. This translates into a 50% wire length increase in the interconnect fabric. We compare our Vdd-programmable FPGA to a single-Vdd FPGA with a Vdd level suggested by ITRS for a 100-nm process. Considering all the area and delay overhead, our Vdd-programmable logic fabric reduces logic power and local interconnect power by 42.30% and 37.97%, respectively, and our Vdd-programmable interconnect fabric reduces global interconnect power by 46.74%. Overall, we are able to reduce the total FPGA power by 47.61%. Our critical path delay is 17.65% larger than that of the single-Vdd FPGA due to the delay overhead of Vdd programmability and longer wire segments. Considering this performance loss, we still reduce the energy-delay product by 27.36%. To the best of our knowledge, it is the first in-depth study on field-programmable supply voltages for FPGA power reduction.

We have studied supply-voltage programmability using the common academic research framework, the generic tile-based architecture presented in [18], and the de facto benchmark suite MCNC benchmark suite [31] for academic research. In the future, we plan to extend the concept of supply-voltage programmability to commercial FPGA architectures and circuits and verify this technique by large industry benchmarks.

Although we have significantly reduced interconnect power, there is still a large amount of leakage power and area overhead due to the level converters inserted in our Vdd-programmable interconnect fabric. Recent studies have reduced such leakage and area overhead. Anderson and Najm [12] use the keeper transistor in the routing structure of commercial FPGAs to perform the level restoring. In essence, it combines the level converter with the routing switch at circuit level. Gayasen *et al.* [11] use the same Vdd level for all the routing trees driven by one output of a logic block. Therefore, the Vdd-level conversion occurs only at the input/output pins of a logic block. With same level-converter placement as in [11], Lin and He [35] customize the Vdd level for each routing tree and further allow high-Vdd wire segments driving low-Vdd segments within each routing tree. It also performs chip-level time slack allocation and obtains significant power reduction compared to both this paper and [11]. The area overhead of Vdd programmability in [34] and [35] is barely 17% for about 54% energy-delay product reduction.

REFERENCES

[1] K. Poon, A. Yan, and S. Wilton, "A flexible power model for FPGAs," in *Proc. 12th Int. Conf. Field-Programmable Logic and Appl.*, Sep. 2002, pp. 312–321.

[2] F. Li, D. Chen, L. He, and J. Cong, "Architecture evaluation for power-efficient FPGAs," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, Feb. 2003, pp. 175–184.

[3] T. Tuan and B. Lai, "Leakage power analysis of a 90 nm FPGA," in *Proc. IEEE Custom Integr. Circuits Conf.*, 2003, pp. 57–60.

[4] J. H. Anderson, F. N. Najm, and T. Tuan, "Active leakage power optimization for FPGAs," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, Feb. 2004, pp. 33–41.

[5] J. Lamoureux and S. J. Wilton, "On the interaction between power-aware FPGA CAD algorithms," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2003, pp. 701–708.

[6] A. Gayasen, Y. Tsai, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and T. Tuan, "Reducing leakage energy in FPGAs using region-constrained placement," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, Feb. 2004, pp. 51–58.

[7] D. E. Lackey *et al.*, "Managing power and performance for system-on-chip designs using voltage islands," in *Proc. Int. Conf. Comput.-Aided Des.*, 2002, pp. 195–202.

[8] F. Li, Y. Lin, L. He, and J. Cong, "Low-power FPGA using predefined dual-Vdd/dual-Vt fabrics," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, Feb. 2004, pp. 42–50.

[9] F. Li, Y. Lin, and L. He, "FPGA power reduction using configurable dual-Vdd," in *Proc. Des. Autom. Conf.*, Jun. 2004, pp. 735–740.

[10] ——, "Vdd programmability to reduce FPGA interconnect power," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2004, pp. 760–765.

[11] A. Gayasen, K. Lee, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, and T. Tuan, "A dual-Vdd low power FPGA architecture," in *Proc. Int. Conf. Field-Programmable Logic and Appl.*, Aug. 2004, pp. 145–147.

[12] J. H. Anderson and F. N. Najm, "Low-power programmable routing circuitry for FPGAs," in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2004, pp. 602–609.

[13] F. Li, Y. Lin, and L. He, "Routing track duplication with fine-grained power-gating for FPGA interconnect power reduction," in *Proc. Asia South Pacific Des. Autom. Conf.*, Jan. 2005, pp. 645–650.

[14] Y. Lin, F. Li, and L. He, "Power modeling and architecture evaluation for FPGA with novel circuits for Vdd programmability," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, Feb. 2005, pp. 199–207.

[15] Y. Lin and L. He, "Leakage efficient chip level dual-Vdd assignment with time slack allocation for FPGA power reduction," in *Proc. Des. Autom. Conf.*, Jun. 2005, pp. 720–725.

[16] D. Chen, J. Cong, F. Li, and L. He, "Low power technology mapping for FPGA architectures with dual supply voltages," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, Feb. 2004, pp. 109–117.

[17] D. Chen and J. Cong, "Delay optimal low-power circuit clustering for FPGAs with dual supply voltages," in *Proc. ACM Int. Symp. Field-Programmable Gate Arrays*, Aug. 2004, pp. 70–73.

[18] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Norwell, MA: Kluwer, Feb. 1999.

[19] G. G. Lemieux and S. D. Brown, "A detailed router for allocating wire segments in field-programmable gate arrays," in *Proc. ACM Phys. Des. Workshop*, Apr. 1993, pp. 215–226.

[20] V. Betz and J. Rose, "VPR: A new packing, placement and routing tool for FPGA research," in *Proc. Int. Workshop Field Programmable Logic and Appl.*, 1997, pp. 213–222.

[21] J. Kao, S. Narendra, and A. Chandrakasan, "MTCMOS hierarchical sizing based on mutual exclusive discharge patterns," in *Proc. Des. Autom. Conf.*, Jun. 1998, pp. 495–500.

[22] U. of Berkeley Device Group, *Berkeley Predictive Technology Model*, 2002. [Online]. Available: http://www-device.eecs.berkeley.edu/ptm/mosfet.html

[23] Y.-M. Jiang, A. Krstic, and K.-T. Cheng, "Estimation for maximum instantaneous current through supply lines for CMOS circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 8, no. 1, pp. 61–73, Feb. 2000.

[24] K. Usami *et al.*, "Automated low-power technique exploiting multiple supply voltages applied to a media processor," *IEEE J. Solid-State Circuits*, vol. 33, no. 3, pp. 463–472, Mar. 1998.

[25] K. Usami and M. Horowitz, "Clustered voltage scaling techniques for low-power design," in *Proc. Int. Symp. Low Power Electron. and Des.*, 1995, pp. 3–8.

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

[27] F. Ishihara, F. Sheikh, and B. Nikolic, "Level conversion for dual-supply systems," in *Proc. Int. Symp. Low Power Electron. and Des.*, 2003, pp. 164–167.

[28] R. Puri, L. Stok, J. Cohn, D. Kung, D. Pan, D. Sylvester, A. Srivastava, and S. Kulkarni, "Pushing ASIC performance in a power envelope," in *Proc. Des. Autom. Conf.*, 2003, pp. 788–793.

[29] J. P. Fishburn and A. E. Dunlop, "TILOS: A posynomial programming approach to transistor sizing," in *Proc. Int. Conf. Comput.-Aided Des.*, 1985, pp. 326–328.

[30] R. W. Brodersen, M. A. Horowitz, D. Markovic, B. Nikolic, and V. Stojanovic, "Methods for true power minimization," in *Proc. Int. Conf. Comput.-Aided Des.*, 2002, pp. 35–42.

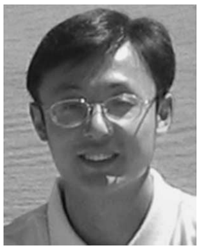
[31] *MCNC Designers' Manual*, MCNC, Research Triangle Park, NC, 1993.

[32] S. Yang, "Logic synthesis and optimization benchmarks," Microelectronics Center North Carolina, Research Triangle Park, NC, Tech. Rep., 1991.

[33] International Technology Roadmap for Semiconductors, 2003. [Online]. Available: http://public.itrs.net/Files/2003ITRS/Home2003.htm

[34] Y. Lin, F. Li, and L. He, "Circuits and architectures for field programmable gate array with configurable supply voltage," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 9, pp. 1035–1047, Sep. 2005.

[35] Y. Lin and L. He, "Leakage efficient chip-level dual-Vdd assignment with time slack allocation for FPGA power reduction," in *Proc. Des. Autom. Conf.*, Jun. 2005, pp. 720–725.



Fei Li received the B.S. and M.S. degrees in electrical engineering from Fudan University, Shanghai, China, in 1997 and 2000, respectively, the M.S. degree in computer engineering from the University of Wisconsin, Madison, in 2002, and the Ph.D. degree in electrical engineering from the University of California, Los Angeles.

His research interests include computer-aided design of VLSI circuits and systems, programmable device architecture, and low-power design.

Yan Lin (S'05) received the B.E. degree in automation from Tsinghua University, Beijing, China, in 2002, and the M.S. degree in electrical engineering from the University of California, Los Angeles (UCLA), Los Angeles, in 2004. He is currently working toward the Ph.D. degree in the Department of Electrical Engineering, UCLA.

His research interests include computer-aided design of VLSI circuits and systems, programmable fabrics, and high-performance and low-power designs.

Lei He received the Ph.D. degree in computer science from the University of California, Los Angeles (UCLA), Los Angeles, in 1999.

He is currently an Associate Professor with the Department of Electrical Engineering, UCLA. From 1999 to 2001, he was a faculty member with the University of Wisconsin, Madison. He also held visiting and consulting positions with Intel, Hewlett-Package, and Synopsys. He has published more than 130 technical papers. His research interests include VLSI circuits and systems, and electronic design automation.

Dr. He was a recipient of the U.S. National Science Foundation CAREER Award in 2000, the UCLA Chancellor's Faculty Career Development Award (highest class) in 2003, the IBM Faculty Award in 2003, the Northrop Grumman Excellence in Teaching Award in 2005, and the Best Paper Award at the 2006 International Symposium on Physical Design. He is a technical program committee member for a number of conferences including the Design Automation Conference, the International Conference on Computer-Aided Design, the International Symposium on Low Power Electronics and Design, and the International Symposium on Field-programmable Gate Array.