Field Programmability of Supply Voltages for FPGA Power Reduction

Fei Li, Yan Lin and Lei He
Electrical Engineering Department
University of California, Los Angeles 90095

Abstract

Power reduction is of growing importance for the field programmable gate array (FPGA). In this paper, we apply programmable supply voltage (Vdd) to reduce power in FPGA. We first design FPGA logic fabrics using dual Vdd levels, and show that field programmable power supply is required to obtain a satisfactory performance and power trade-off. We further design FPGA interconnect fabrics with fine-grained Vdd programmability but minimal increase of configuration SRAM cells. With a simple yet practical CAD flow to leverage the field programmable dual-Vdd logic and interconnect fabrics, we carry out a highly quantitative study using placed and routed benchmark applications and delay/power models obtained from detailed circuit designs in 100nm technology. Compared to single-Vdd FPGAs with Vdd level suggested by ITRS for 100nm technology, field programmable dual-Vdd FPGA reduces total power by 33.37% and energy-delay product by 28.97%. To the best of our knowledge, it is the first in-depth study on programmable Vdd for power reduction in FPGA.

Index Terms

supply voltage programmability, dual Vdd, FPGA architecture, power reduction.

This paper is partially supported by NSF CAREER award CCR-0401682 and NSF grant CCR-0306682. We used computers donated by Intel and SUN Microsystems. Address comments to lhe@ee.ucla.edu.

Field Programmability of Supply Voltages for FPGA Power Reduction

I. Introduction

FPGA is an attractive design platform due to its low NRE (non-recurring engineering) cost and the short time-to-market. However, the power efficiency of FPGAs is much lower than ASIC because a large number of transistors are used for field programmability and the utilization rate of FPGA resources is intrinsically low. FPGA power modeling and analysis have drawn growing attention. [1] [2] present flexible power models for generic parameterized FPGA architectures and show that both interconnect and leakage power are significant power components for existing FPGAs. [3] analyzes the leakage power of a commercial FPGA architecture in 90nm technology and quantifies the leakage power challenge for nanometer FPGAs. As power consumption becomes the increasingly important design constraint for FPGAs, FPGA power reduction has also been studied recently. [4] introduces an inversion method to re-program the FPGA configuration bits and reduces leakage power of multiplexers without additional hardware cost. [5] develops a suite of power-aware CAD algorithms for existing FPGA architectures. Other work involves designing power-efficient FPGA circuits. For example, [6] investigates power-gating of logic fabrics and applies region-constrained placement to reduce leakage power of unused logic blocks.

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

The rest of the paper is organized as follows. Section II presents background knowledge and our baseline FPGA architecture. Section III introduces the field programmable power supply to logic fabrics and discusses the detailed circuit and fabric design. Section IV presents the CAD flow with consideration of Vdd-programmable logic fabrics. Experiments in Section V show that the programmable dual-Vdd can reduce logic and local interconnect power by 42.30% and 37.97% respectively. 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. Experimental results show that the Vdd-programmable logic and global interconnects together can reduce total FPGA power by 33.37% and reduce energy-delay product by 28.97%. We conclude this paper in Section VII. To the best of our knowledge, this paper and associated conference papers present the first in-depth study of field programmable Vdd for FPGAs[1].

II. Background and Preliminaries

A. Cluster-based Island Style FPGAs

This paper assumes cluster-based island style FPGA architectures, which are used for most commercial FPGAs [11] [12]. Figure 1 presents a cluster-based logic block and basic logic elements [13]. A logic block is a cluster of fully connected Basic Logic Elements (BLEs), and the cluster size is the number of BLEs in a logic block. Each BLE consists of one Lookup Table (LUT) and one flip-flop. Figure 2 shows the schematic of a 4-input LUT, which includes SRAM cells and a multiplexer tree. The SRAM cells in the LUT can be programmed to implement any four-input logic function. The island style routing structure is shown in Figure 3 (a). Logic blocks are surrounded by programmable routing channels and routing wires in both horizontal and vertical channels are segmented by *routing switch blocks*. Figure 3 (c) shows a subset switch block [14], where the incoming track can be connected to the outgoing tracks with the same track number. The connections in a switch block (represented by the dashed lines in Figure 3 (c)) are programmable routing switches. Routing switches can be implemented by tri-state buffers and each connection needs two tri-state buffers so that it can be programmed independently for either direction[2].

[1]We first introduced the concept of Vdd programmability in [8] and then reported the preliminary studies on Vdd programmable logic and interconnect fabrics in [9] [10], respectively.

[2]Modern FPGAs also use unidirectional routing switches. Without loss of generality, we assume bi-directional switches that are implemented by tri-state buffers but our low-power techniques also applies to uni-directional switches.

In this paper, we assume that the LUT size is 4, cluster size is 10, and the interconnect structure contains 100% tri-state buffers (rather than a mix of buffers and pass transistors). We customize 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 [13], i.e., $W = 1.2W_{min}$, where W_{min} is the minimum channel width required to route the given circuit successfully. The channel width W represents a "low-stress" routing situation that usually occurs in commercial FPGAs for 'average' circuits. Similar to [13], we conduct experiments by placing and routing MCNC benchmark set in 100nm technology.

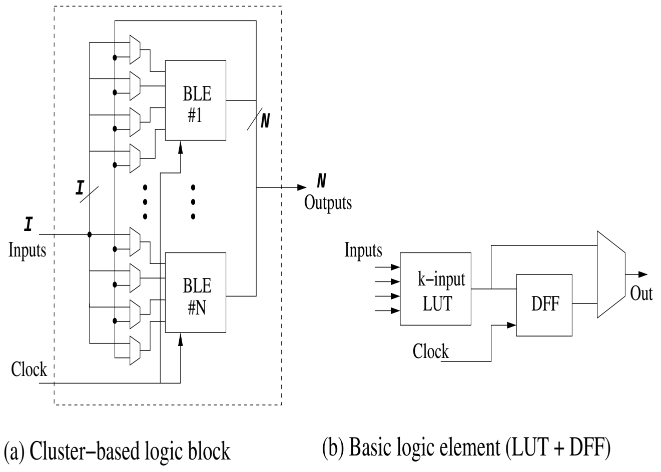

BLE
#1
N
I
N
Outputs
I
Inputs
BLE
#N
Clock
Inputs
k-input
LUT
DFF
Out
Clock
(a) Cluster-based logic block
(b) Basic logic element (LUT + DFF)

Fig. 1. FPGA logic block and basic logic element.

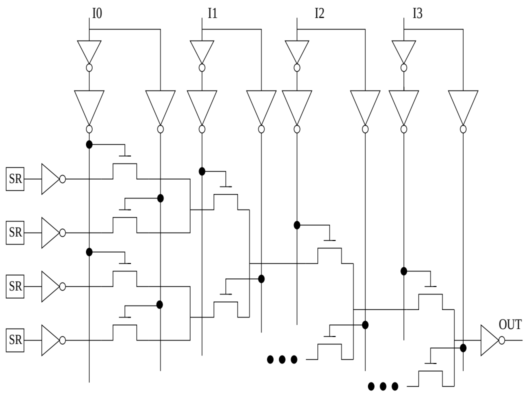


Fig. 2. The schematic of a 4-input LUT (SR stands for SRAM cell).

B. Low Leakage 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 interconnect 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]. Figure 4 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, and the rest part 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 introducing runtime delay penalty. In reality, a too high Vt increases the SRAM write time (i.e., FPGA configuration time) significantly. In this paper, we decide to increase the SRAM cell Vt for 15X SRAM leakage reduction while only increasing the configuration time by 13%. This trade-off is justifiable because the configuration time is not critical in most FPGA applications. Note that the high-Vt low-leakage SRAM cells can also be used for the programmability of interconnects. In the rest of the paper, we assume the low-leakage SRAMs are used for all the programmability in FPGAs.

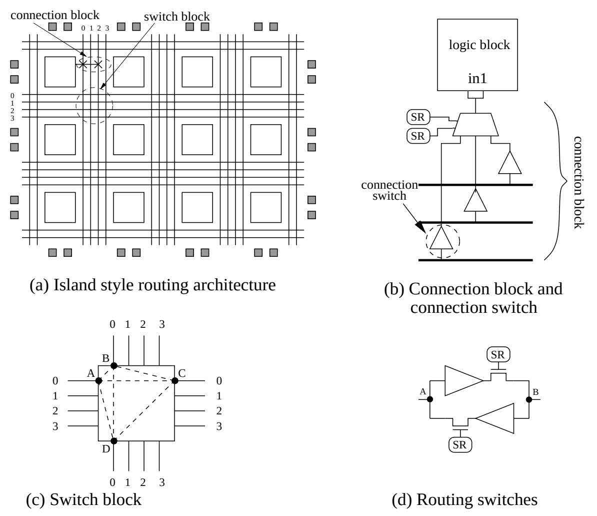


Fig. 3. (a) Island style routing architecture; (b) Connection block; (c) Switch block; (d) Routing switches.

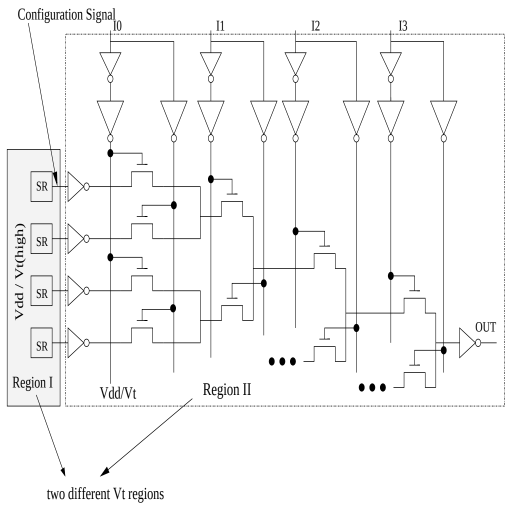


Fig. 4. The schematic of a 4-input LUT using high-Vt (low-leakage) SRAM cells (SR stands for SRAM cell).

III. Vdd Programmable Logic Fabrics

To introduce dual-Vdd and Vdd programmability to FPGA logic fabrics, we design the detailed circuits and fabrics in this section. We also discuss the corresponding CAD flow to leverage the new FPGA logic fabrics. We assume that the global interconnects use uniform VddH in this section.

A. Circuit Design for Logic Blocks

We design three types of logic blocks as shown in Figure 5. The first two types, *H-block* and *L-block*, are connected to supply voltages VddH and VddL respectively. A H-block has the highest performance but a L-block has reduced power consumption at the cost of the increased delay. To further introduce the supply voltage programmability, a Vdd-programmable logic block, named as *P-block* in Figure 5 (d) is designed. We implement a P-block by inserting PMOS transistors between the power supply rails and the logic block. These transistors are called *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 for power gating [15]. An important design aspect for sleep transistor is how to determine its size because it impacts both performance and area overhead. In our circuit design, 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 the specified performance. We design power switches with normal Vt so that area overhead can be reduced. Figure 6 presents the SPICE simulation results for a 4-input LUT with power switch in 100nm technology. The X-axis is the power switch area in the percentage of original 4-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.

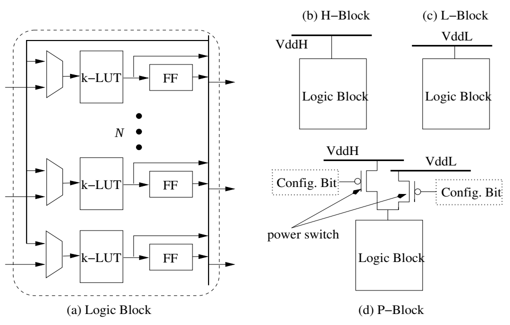


Fig. 5. Logic blocks in dual-Vdd and Vdd-programmable FPGAs.

Clearly, for a same area overhead, a power switch with normal Vt has smaller delay compared to a power switch with high Vt. Our simulation shows that, compared to a normal 4-input LUT at the same Vdd level, an optimized 4-input LUT with power switches has 5% extra delay and 21% transistor area overhead.

To further reduce the power switch area, we make use of the fact that peak current for different parts of a circuit normally do not occur at the same time [15]. We insert power switches for each logic block and then carry out SPICE simulation. For a logic block with cluster size of 10, only 12% area overhead is required to achieve the same 5% performance loss for the worst-case simultaneous switching current. Therefore, large granularity significantly reduces the power switch size and the transistor area overhead. In this paper, we decide to insert power switches at the logic block level. To select different supply voltages, we need two power switches for each logic block. According to Figure 6, different supply voltages almost do not change the area overhead in order to get the same delay increase. To limit the delay increase to 5%, a P-block logic cluster requires 24% area overhead for two power switches, again for the worst-case simultaneous switching current.

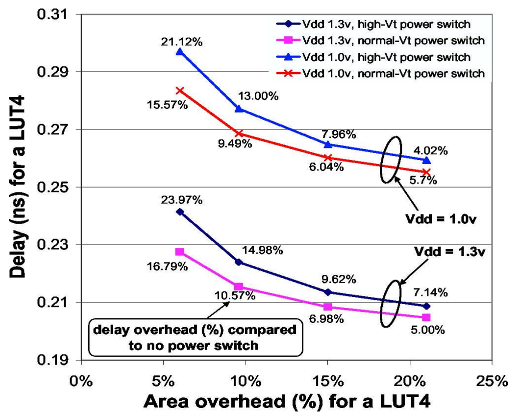


Fig. 6. Area and delay overhead of the power switch for a 4-input LUT.

The configuration SRAM cells to control the power switches lead to extra leakage power. But 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 happens on the source/drain capacitors. Therefore, the major power overhead of Vdd programmability is due to the configuration SRAM cells. Because high-Vt SRAM can be applied to all configuration SRAM cells, the increased number of SRAM cells due to supply voltage programmability not necessarily leads to a large leakage power increase.

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. Note that gate-boosting has already been used in some commercial Xilinx FPGAs [13] to compensate the logic ‘1’ degradation of NMOS pass transistors in routing switches. Therefore, it is not difficult to implement gate-boosting for our PMOS power switches.

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

TABLE I
LEAKAGE POWER FOR A P-BLOCK CONTAINING ONE 4-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

NMOS transistors in the VddH device will be in partially "on" state, and dissipate 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 VddL signal swing to VddH signal swing [3]. Different level converter circuits have been used in dual-Vdd ASIC designs [16], [17], [18], [19]. We use the recently proposed asynchronous level converters with single supply voltage [20] in our dual-Vdd fabrics. Figure 7 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 will 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 (p2,n2) generates a VddH signal to the second inverter so that no DC short circuit current exists. For a particular VddH/VddL combination, we decide the transistor size in

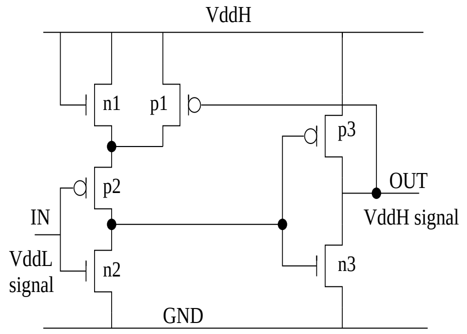


Fig. 7. A level converter circuit with single supply voltage.

the level converter as follows. We start from a level converter with minimum transistor sizes. We 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 a proper virtual low-supply as 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.

TABLE II
DELAY AND POWER OF THE LEVEL CONVERTER IN FIGURE 7 AT BERKLEY PREDICTED 100NM 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

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, pre-defined dual-Vdd fabric and Vdd-programmable fabric, in the following.

1) Pre-defined Dual-Vdd Fabric: The pre-defined dual-Vdd fabric, named as *arch-DV* in this paper, is a mixture of H-blocks and L-blocks. There is a pre-defined 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. Figure 8 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.

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

Note that the routing resources use uniform VddH because this section focuses on applying dual Vdd only to logic blocks. The advantage of dual-Vdd routing fabric is explored in Section VI. Figure 8 also shows example routing paths connecting 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 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.

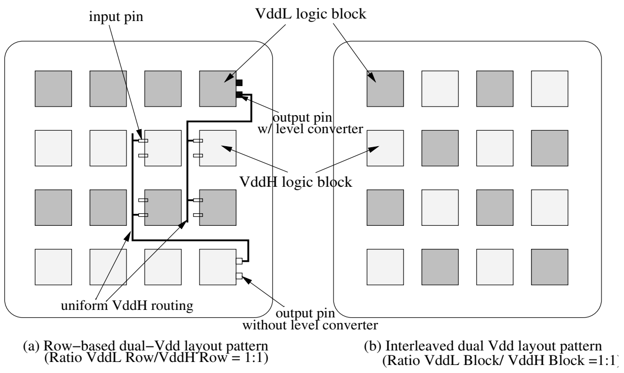

input pin
VddL logic block
output pin
w/ level converter
VddH logic block
uniform VddH routing
output pin
without level converter
(a) Row−based dual−Vdd layout pattern
(Ratio VddL Row/VddH Row = 1:1)
(b) Interleaved dual Vdd layout pattern
(Ratio VddL Block/ VddH Block =1:1

Fig. 8. Pre-defined dual-Vdd layout patterns.

2) Vdd-Programmable Logic Fabric: When all the logic blocks in a 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 the three types of logic blocks and it has the level of Vdd programmability between the pre-defined 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 experiments on fabrics of mixed H-block/L-block/P-block presented in [9].

IV. CAD Flow for Vdd-Programmable Logic Fabric

We develop corresponding CAD algorithms and design flow in Figure 9 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 [13] to obtain a cluster-level netlist. We then perform single-Vdd timing-driven placement and routing by VPR [13] and generate the back-annotated basic circuit netlist (BC-netlist) defined in [8]. As the first step to consider dual Vdd in our design flow, Vdd assignment for logic blocks is performed to obtain a dual-Vdd BC-netlist[4]. 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 FPGA power analysis framework *fpgaEva-LP* [2] [21] is used to estimate the power and performance. If the logic fabric is a pre-defined dual-Vdd fabric, the corresponding design path goes through additional steps of dual-Vdd placement. We discuss the dual-Vdd assignment and dual-Vdd placement in the following 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 surplus timing slack in a circuit and performs power optimization by using dual Vdd levels. Sensitivity-based algorithms have been used in ASIC circuit tuning either for delay optimization [22] or for power-delay trade-off [23]. We use a similar sensitivity-based algorithm for dual-Vdd assignment. First, we define the *power sensitivity* as follows,

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

$$\begin{aligned} S_x &= \frac{\Delta P}{\Delta x} \\ &= \frac{\Delta P_{sw}}{\Delta x} + \frac{\Delta P_{lkg}}{\Delta x} \end{aligned}$$

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

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

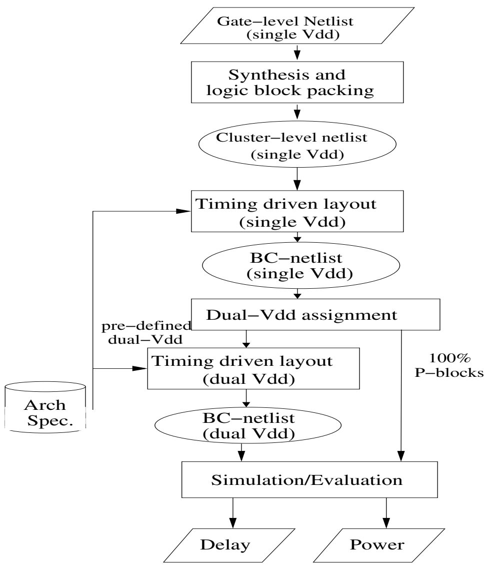

Gate–level Netlist (single Vdd)
Synthesis and logic block packing
Cluster–level netlist (single Vdd)
Timing driven layout (single Vdd)
BC–netlist (single Vdd)
Dual–Vdd assignment
pre–defined dual–Vdd
Timing driven layout (dual Vdd)
100% P–blocks
Arch Spec.
BC–netlist (dual Vdd)
Simulation/Evaluation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

Fig. 9. Design flow for pre-defined dual-Vdd and Vdd-programmable logic fabrics.

In our dual-Vdd assignment problem, the design variable x becomes supply voltage Vdd. To calculate the 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,

$$P_{sw}(i) = 0.5f \cdot \hat{E}_i \cdot C_i \cdot V_{dd}^2 \tag{1}$$

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,

$$P_{lkg}(i) = I_{lkg}(V_{dd}) \cdot V_{dd} \tag{2}$$

where I_{lkg} is the leakage current at supply voltage V_{dd}. 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.

$$S_x(B) = \sum_{node\ i \in B} S_x(i) \tag{3}$$

We present our dual-Vdd assignment algorithm in Figure 10. 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 timing slack. Power sensitivity is calculated for logic blocks on this path but not on the critical path. We select the logic block with the largest power sensitivity and assign low Vdd to it, 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 next iteration. In either case, the logic block selected in current iteration will not be re-visited in other iterations. Right after the dual-Vdd assignment, we can estimate the power and delay for the dual-Vdd BC-netlist as shown in Figure 9. However, this dual-Vdd BC-netlist does not consider the logic cell placement constraint imposed by the pre-defined dual-Vdd pattern. It assumes the flexibility to assign low-Vdd to a logic block at arbitrary physical location. We call it *ideal case* for fabric arch-DV. To obtain *real case* power and delay considering the layout pattern constraint, we use this dual-Vdd netlist as an input and perform dual-Vdd placement.

B. Placement for Dual-Vdd and Vdd Programmable Fabrics

We develop dual-Vdd placement algorithm based on the simulated annealing algorithm implemented in VPR [13]. 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, which is shown as follows,

$$Cost_{lin-cgst} = \sum_{i=1}^{N_{nets}} q(i)[\frac{bb_x(i)}{C_{av,x}(i)^{\beta}} + \frac{bb_y(i)}{C_{av,y}(i)^{\beta}}] \tag{4}$$

The summation is over the number of nets N_{nets} in the circuit. For each net i, $bb_x(i)$ and $bb_y(i)$ represents the horizontal and vertical spans of its bounding box, respectively. The $q(i)$ compensating factor is due to the fact that the wire length model using

```
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 partially assigned BC-netlist $N_p$ be input netlist $N$;
While( $N_p$ has logic blocks not tried )
begin
    Find path $p$ with largest timing slack in $N_p$;
    Get logic blocks on path $p$ but not on critical path;
    Calculate power-sensitivity for those logic blocks;
    Select logic block $B$ with largest sensitivity;
    Assign low Vdd to $B$ and update timing information;
    If( delay constraint not met )
    begin
        Reverse the low-Vdd assignment;
    end
    mark logic block $B$ as 'tried';
end
Let the output netlist $N'$ be $N_p$
```

Fig. 10. Sensitivity-based dual-Vdd assignment algorithm.

bounding box underestimates the wiring required to connect nets with more than three terminal. The value of $q(i)$ depends on the number of terminals in net i. $C_{av,x(i)}$ and $C_{av,y(i)}$ are the average channel capacities in x and y directions, respectively, over the bounding box of net i. When the channel capacities are different across the 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 in our dual-Vdd placement. We define *moves* 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 is

$$\begin{aligned} \Delta Cost(relocate) &= \Delta Cost_{lin-cgst} + \alpha \cdot \Delta matched(j) \\ &\quad + \gamma \cdot (1 - matched(j)) \\ &\quad + \beta \cdot \Delta prog(j) + \theta \cdot prog(j) \end{aligned} \quad (5)$$

$matched(j)$ is a Boolean function describing the Vdd-matching state of a logic block in the new slot and is defined as

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

If the Vdd assigned to block j matches the Vdd at its physical location, $matched(j)$ returns value '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. $\Delta matched(j)$ is the difference of $match(j)$ to penalize 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. 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 a H-block or L-block slot. Similar to $matched(j)$, $prog(j)$ is the Boolean function that designates whether the current location of block j is a P-block slot or not. The term $\Delta prog(j)$ penalizes relocating block j from a Vdd non-programmable slot to a Vdd programmable slot, and the term $prog(j)$ penalizes relocating block j from a Vdd programmable slot to another Vdd programmable slot. Weights α, β, γ and θ are determined experimentally. The cost of swapping two logic blocks is the sum of the costs given by Equation 5 for the two blocks.

V. Experimental Results for Vdd-Programmable Logic Fabrics

In this section, we first compare four types of FPGAs: arch-SV, arch-DV, arch-PV and ideal-DV. The difference between the three FPGAs lies in the logic fabric. Their interconnect fabrics all use uniform Vdd. 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 pre-defined 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 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. For all the four FPGAs, we use LUT size 4 and logic block size 10 in our experiments.

Before we present the experimental results, we need to determine the ratio between H-block and L-block for pre-defined dual-Vdd FPGA *arch-DV*. Table III shows the percentage of VddL logic blocks after dual-Vdd assignment for 20 benchmark circuits. No delay-increase is allowed when we assign VddL to logic blocks. VddH and VddL are set to 1.3v and 0.8v, respectively. The percentage varies from 53% to 96% and the average is around 75%. It clearly shows that circuits implemented on the uniform-Vdd FPGA have a large amount of surplus timing slack to be utilized for power reduction. According to the ideal percentage given by dual-Vdd assignment and considering that the pre-defined 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].

TABLE III

PERCENTAGE OF VDDL LOGIC BLOCKS GIVEN BY DUAL-VDD WITH ZERO DELAY-INCREASE AND NO LAYOUT RESTRICTIONS. (VDDH = 1.3V AND VDDL = 0.8V)

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

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 Figure 8 have been tried for arch-DV. However, our experimental results show no significant power and performance difference between these two layout patterns. Considering that row-based layout pattern is easier to route the power/ground network, we only present the experimental results of row-based layout pattern for arch-DV.

Figure 11 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 represent the power vs. 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 a poor performance for the pre-defined dual-Vdd FPGA. The placement constraint for pre-defined dual-Vdd fabric arch-DV is large enough to degrade the performance dramatically and equivalently leads to large power overhead at the same clock frequency.

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

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

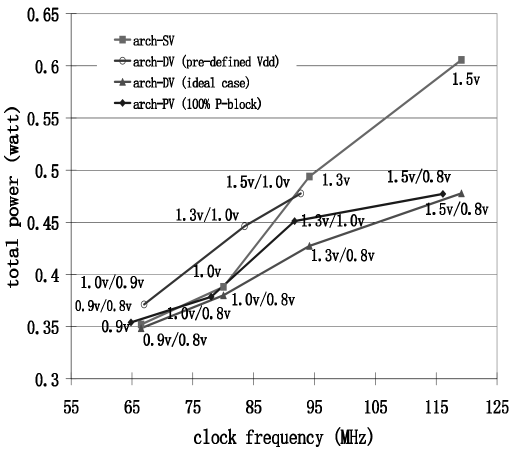


Fig. 11. Power versus delay for *s38584* (arch-SV: single-Vdd FPGA; arch-DV: FPGA with pre-defined dual-Vdd logic fabric; arch-PV: FPGA with Vdd-Programmable logic fabric).

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 performance tradeoff curve compared to arch-SV (See Figure 11). The advantage of FPGA architecture arch-PV over arch-DV has been observed for all the benchmark circuits, and it shows that field programmability of Vdd is required to obtain a satisfactory performance versus power tradeoff when dual Vdd is used to reduce FPGA power. 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 more surplus timing slack can be utilized at the logic block level. Moreover, FPGA arch-PV gives a power performance tradeoff curve which 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

Power and delay comparison between FPGA *arch-PV* (with 100% P-blocks) and the baseline FPGA *arch-SV*. The Vdd is 1.3v for arch-SV and VddH/VddL combination is 1.3v/0.8v 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%

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 [24]. The Vdd level for arch-SV is 1.3v as suggested by ITRS roadmap [25] for 100nm technology. The VddH/VddL combination for arch-PV is 1.3v/0.8v. 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 breakdown FPGA power into logic power, local interconnect power and global interconnect power. The logic power is the power of LUTs, flip-flops and MUXes 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 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 consumes most of the power in an FPGA. As shown by the power breakdown in Table IV for arch-SV, 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 Figure 12 (a). The right part of the circuit is the Vdd-programmable routing switch. For the tri-state buffer in the routing switch, we insert two PMOS transistors M3 and M4 between the tri-state buffer and VddH, VddL power rails respectively. Similar to a Vdd-programmable logic block, turning off one of the two power 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: high Vdd, low Vdd and power-gating.

The power gating state provided by Vdd programmability is very attractive because our SPICE simulation shows that 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 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. Further, the Vdd-programmable routing switch has an increased delay compared to the conventional routing switch because the power switches are inserted between the buffer and power supply. We properly size the power switches for the tri-state buffer to achieve a bounded delay increase. For a routing architecture with all wire segments spanning four logic blocks, we assume 7X minimum width tri-state buffers and get 16% delay increase by inserting 4X minimum width power switches. The left part of the circuit in Figure 12 (a) is the level converter. We insert the level converter right before the routing switch and use a multiplexer to either use this level converter or bypass it. The transistor M1 is used to prevent signal transitions from propagating through the level converter when it is bypassed, and 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 due to the transistor M1 and the multiplexer is modeled in our CAD flow.

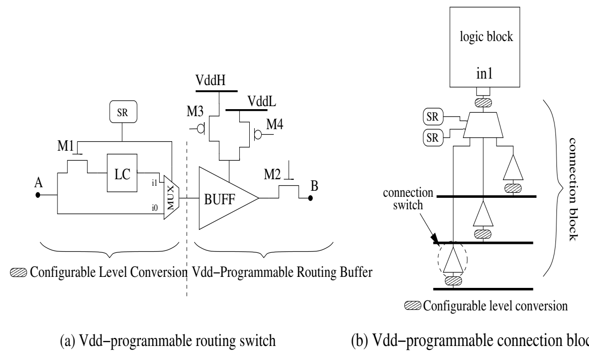

SR
M1
LC
A
i1
i0
MUX
VddH
VddL
M3
M4
M2
BUFF
B
Configurable Level Conversion
Vdd–Programmable Routing Buffer
logic block
in1
SR
SR
connection switch
connection block
Configurable level conversion
(a) Vdd–programmable routing switch
(b) Vdd–programmable connection block

Fig. 12. (a) Vdd-programmable routing switch; (b) Multiplex-based Vdd-programmable connection block. (SR stands for SRAM cell and LC stands for level converter. The same configurable level conversion circuit is used in both (a) and (b))

Another type of routing resources are the connection blocks [13]. Figure 12 (b) shows the multiplexer-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 multiplexer are connection switches. To apply Vdd programmability to connection blocks, we can simple 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 $\lceil log_2 N \rceil$ SRAM cells are for the multiplexer and the other $2N$ SRAM cells are for 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

[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.

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 V

UTILIZATION RATE OF INTERCONNECT SWITCHES.

Vdd	routing switch delay (ns)		energy per switch (Joule)	
	without Vdd programmability	with Vdd programmability (increase %)	without Vdd programmability	with Vdd programmability
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

TABLE VI

DELAY AND POWER OF A VDD-PROGRAMMABLE ROUTING SWITCH. WE USE 7X MINIMUM WIDTH TRI-STATE BUFFER FOR ROUTING SWITCHES AND 4X MINIMUM WIDTH PMOS TRANSISTOR FOR POWER SWITCHES.

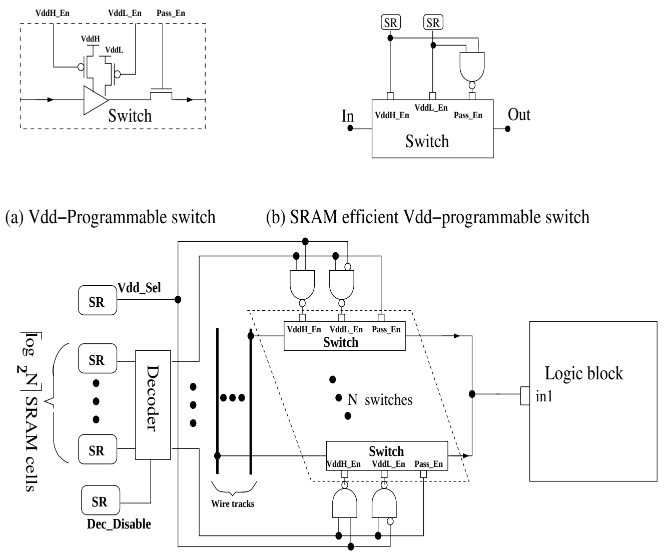
proposed in [21] to improve the SRAM efficiency as well as the delay. As shown in Figure 13 (a), a Vdd-programmable switch module with three signal ports, $VddH\_En$, $VddL\_En$ and $Pass\_En$, is first defined. By setting these three control signals, we can program the Vdd-programmable switch between Vdd selection and power-gating. Figure 13 (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 Figure 13 (c). It removes the multiplexer and tie the tri-state 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. This design reduces the number of SRAM cells for a N-input connection block to $\lceil log_2 N\rceil + 2$. Further, the new connection block is 28% faster than the multiplexer-based connection block and 19% less dynamic power as shown in Table VII. In this paper, 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 this type of connection, we also insert the configurable level conversion circuit before each logic block input pin.

Vdd	connection switch 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%)

TABLE VII

THE DELAY AND POWER OF THE SRAM-EFFICIENT VDD-PROGRAMMABLE CONNECTION BLOCK. WE USE 4X MINIMUM WIDTH TRI-STATE BUFFER FOR CONNECTION SWITCHES AND 1X MINIMUM WIDTH PMOS TRANSISTOR FOR POWER TRANSISTORS.

The resulting FPGA has Vdd-programmability for both logic and interconnect fabrics and we name it as *arch-PV-fpga*. The same design path for FPGA arch-PV (with 100% P-blocks) in Figure 9 can be applied to arch-PV-fpga. The only change is that a circuit element in the step of dual-Vdd assignment can be either a logic block or an interconnect switch. Power sensitivity is calculated for both logic blocks and interconnect switches.



(a) Vdd−Programmable switch (b) SRAM efficient Vdd−programmable switch

(c) SRAM−efficientVdd programmable connection block

Fig. 13. (a) Vdd-programmable switch (b) SRAM-efficient Vdd-programmable switch; (c) SRAM-efficient Vdd-programmable connection block.

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. As shown in Column 6 - 8, the leakage power of global interconnects is reduced by 23.20% and the dynamic power of global interconnects is reduced by 40.89%. The overall global interconnect power is reduced by 28.89%. With this power reduction for global interconnects, arch-PV-fpga is able to reduce total FPGA power by 33.37%. 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). Although the circuit level delay increase for a Vdd-programmable interconnect switch is 16%, the SARM efficient connection block is 28% faster compared to the multiplexer-based connection block. Therefore, at system level, arch-PV-fpga only has a small delay increase of 3.24% compared arch-SV. Considering this performance loss, we compare the metric of energy-delay product in Column 10 and show that arch-PV-fpga can still reduce energy-delay product significantly (28.97% on average for the MCNC benchmark circuits).

TABLE VIII

POWER AND DELAY COMPARISON BETWEEN FPGA *arch-PV-fpga* AND BASELINE FPGA *arch-SV*. THE VDD IS 1.3V FOR ARCH-SV AND VDDH/VDDL COMBINATION IS 1.3V/0.8V FOR ARCH-PV-FPGA.

1	2	3	4	5	6	7	8	9	10
circuit	arch-SV (baseline)			arch-PV-fpga					
	delay (ns)	global intcnt. power (Watt)	total power (Watt)	delay increase	global intcnt. power saving			total FPGA power saving	FPGA energy-delay product reduction
					overall	leakage	dynamic		
alu4	10.38	0.0974	0.1450	2.13%	29.18%	22.58%	35.18%	36.78%	31.47%
apex2	11.10	0.1528	0.2040	4.11%	30.25%	22.82%	37.63%	32.85%	27.96%
apex4	10.13	0.0807	0.1056	3.57%	25.56%	23.18%	29.85%	30.14%	25.78%
bigkey	6.34	0.2267	0.3658	3.07%	32.15%	22.02%	41.45%	42.84%	36.93%
clma	21.37	0.9147	1.0172	5.04%	24.41%	22.02%	37.69%	28.74%	21.38%
des	10.60	0.3183	0.4251	5.04%	34.63%	24.16%	44.85%	40.01%	37.20%
diffeq	11.93	0.0580	0.0726	2.32%	28.27%	24.78%	47.68%	30.59%	29.73%
dsip	5.49	0.2635	0.3634	0.62%	34.02%	23.39%	45.25%	37.59%	33.03%
elliptic	16.02	0.2046	0.2395	3.58%	27.98%	22.31%	52.52%	32.72%	26.76%
ex1010	14.69	0.2818	0.3312	4.34%	23.35%	21.02%	31.85%	29.87%	20.94%
ex5p	11.49	0.0772	0.0973	6.18%	26.55%	23.41%	33.96%	30.13%	26.09%
frisc	22.30	0.3188	0.3515	2.85%	27.15%	25.57%	46.29%	28.06%	27.58%
misex3	9.65	0.1092	0.1514	0.33%	29.92%	23.65%	35.22%	33.26%	29.91%
pdc	14.67	0.4234	0.4859	2.48%	26.01%	22.09%	40.12%	30.83%	23.83%
s298	21.16	0.0813	0.1069	4.94%	30.95%	23.60%	50.49%	35.29%	31.54%
s38417	14.63	0.3842	0.4791	2.86%	26.22%	23.21%	35.18%	30.55%	26.14%
s38584	10.62	0.3700	0.4923	3.12%	35.37%	24.67%	52.84%	39.95%	38.70%
seq	9.31	0.1521	0.2061	1.03%	29.84%	23.56%	36.14%	32.18%	28.35%
spla	13.77	0.2606	0.3089	2.79%	27.40%	21.41%	42.89%	33.59%	25.73%
tseng	12.47	0.0450	0.0605	5.76%	28.54%	24.51%	40.69%	31.46%	30.26%
Avg.	-	-	-	3.24%	28.89%	23.20%	40.89%	33.37%	28.97%

VII. Conclusions and Discussions

We have proposed supply voltage (Vdd) programmability to reduce FPGA power. We first design FPGA logic fabrics using dual Vdd levels to reduce dynamic power and show that field programmable Vdd is required to achieve a satisfactory power and performance trade-off. With a simple yet practical CAD flow to leverage the new dual-Vdd logic fabrics, we carry out a highly quantitative study using placed and routed benchmark circuits and area, delay and power models obtained from detailed circuit design in 100nm technology. Compared to single-Vdd FPGA with Vdd level suggested by ITRS for 100nm process, our Vdd programmable logic fabric reduces logic power and local interconnect power by 42.30% and 37.97% respectively. Our Vdd programmable interconnect fabric reduces global interconnect power by 28.89%. With the power savings for logic and interconnect fabric together, the total FPGA power is reduced by 33.37%. Due to delay overhead of Vdd programmability at circuit level, the critical path delay of our Vdd programmable FPGA is 3.24% larger compared to single-Vdd FPGA. With such small performance degradation, Vdd programmable FPGA significantly reduces energy-delay product by 28.97%. To the best of our knowledge, it is the first in-depth study on programmable Vdd for FPGA power reduction.

Although we have significantly reduced FPGA interconnect power, there is still a large amount of leakage power for the level converters inserted in our fine-grained Vdd-programmable interconnect fabric. Our recent study has significantly reduced the number of level converters [26] and conducted architecture evaluation with Vdd programmability [21]. Power supply network to support configurable Vdd or dual-Vdd may introduce extra routing congestion. Leveraging our recent research on optimal synthesis of sleep transistors and power supply network [27], [28], we will study power delivery design and optimization for Vdd programmable FPGAs in the future.

References

[1] K. Poon, A. Yan, and S. Wilton, "A flexible power model for FPGAs," in *Proc. of 12th International conference on Field-Programmable Logic and Applications*, Sep 2002.

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

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

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

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

[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 Intl. Symp. Field-Programmable Gate Arrays*, February 2004.

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

[8] F. Li, Y. Lin, L. He, and J. Cong, "Low-power FPGA using pre-defined dual-vdd/dual-vt fabrics," in *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, Februray 2004.

[9] F. Li, Y. Lin, and L. He, "FPGA power reduction using configurable dual-vdd," in *Proc. Design Automation Conf.*, June 2004, pp. 735–740.

[10] Fei Li and Yan Lin and Lei He, "Vdd programmability to reduce fpga interconnect power," in *Proc. Intl. Conf. Computer-Aided Design*, November 2004, pp. 760–765.

[11] Xilinx Corporation, "Virtex-II 1.5v platform FPGA complete data sheet," July 2002.

[12] Altera Corporation, "Stratix programmable logic device family data sheet," Aug 2002.

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

[14] G. G. Lemieux and S. D. Brown, "A detailed router for allocating wire segments in field-programmable gate arrays," in *Proceedings of the ACM Physical Design Workshop*, April 1993.

[15] J. Kao, S. Narendra, and A. Chandrakasan, "MTCMOS hierarchical sizing based on mutual exclusive discharge patterns," in *Proc. Design Automation Conf.*, June 1998, pp. 495–500.

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

[17] K. Usami and M. Horowitz, "Clustered voltage scaling techniques for low-power design," in *Proc. Intl. Symp. Low Power Electronics and Design*, 1995.

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

[19] F. Ishihara, F. Sheikh, and B. Nikolic, "Level conversion for dual-supply systems," in *Proc. Intl. Symp. Low Power Electronics and Design*, 2003, pp. 164 – 167.

[20] 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. Design Automation Conf.*, 2003, pp. 788 – 793.

[21] Y. Lin, F. Li, and L. He, "Power modeling and architecture evaluation for FPGA with novel circuits for vdd programmability," in *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, Februray 2005.

[22] J. P. Fishburn and A. E. Dunlop, "TILOS: A posynomial programming approach to transistor sizing," in *Proc. Intl. Conf. Computer-Aided Design*, 1985, pp. 326–328.

[23] R. W. Brodersen, M. A. Horowitz, D. Markovic, B. Nikolic, and V. Stojanovic, "Methods for ture power minimization," in *Proc. Intl. Conf. Computer-Aided Design*, 2002, pp. 35–42.

[24] S. Yang, "Logic synthesis and optimization benchmarks," Microelectronics Center of North Carolina, Tech. Rep., 1991.

[25] International Technology Roadmap for Semiconductors, in *2003 Edition, http://public.itrs.net/Files/2003ITRS/Home2003.htm*, 2003.

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

[27] C. Long and L. He, "Distributed sleep transistor network for leakage power reduction," in *Proc. Design Automation Conf.*, June 2003.

[28] C. Long, J. Xiong, and L. He, "On optimal physical synthesis of sleep transistors," in *Proc. Intl. Symp. Physical Design*, April 2004.