Power Modeling and Architecture Evaluation for FPGA with Novel Circuits for Vdd Programmability *

Yan Lin
Electrical Engineering Department
University of California, Los Angeles

Fei Li
Electrical Engineering Department
University of California, Los Angeles

Lei He
Electrical Engineering Department
University of California, Los Angeles

ABSTRACT

Vdd-programmable FPGAs have been proposed recently to reduce FPGA power, where Vdd levels can be customized for different circuit elements and unused circuit elements can be power-gated. In this paper, we first develop an accurate FPGA power model and then design novel Vdd-programmable interconnect switches with minimum number of configuration SRAM cells. Applying our power model to placed and routed benchmark circuits, we evaluate Vdd-programmable FPGA architecture using the new switches. The best architecture in our study uses Vdd-programmable logic blocks and Vdd-gateable interconnects. Compared to the baseline architecture similar to the leading commercial architecture, the best architecture reduces the minimal energy-delay product by 44.14% with 48% area overhead and 3% SRAM cell increase. Our evaluation results also show that LUT size 4 always gives the lowest energy consumption while LUT size 7 always leads to the highest performance for all evaluated architectures.

Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]: Types and Design Styles

General Terms

Design, Algorithms

Keywords

FPGA architecture, FPGA power model, low power, dual-Vdd, Vdd programmability

1. INTRODUCTION

FPGA provides an attractive design platform with low NRE (non-recurring engineering) cost and short time-to-market. Due to a large number of transistors for field programmability and low utilization rate of FPGA resources, existing FPGAs are highly power hungry compared to ASICs. Previous study [1] has shown a 100x energy difference between FPGA designs and their ASIC counterparts. As the process advances to nanometer technology and low-energy embedded applications are explored for FPGAs, power consumption becomes a crucial design constraint for FPGAs. Some recent work has studied FPGA power modeling and optimization. [2, 3] present power evaluation frameworks for generic parameterized FPGA architectures and show that both interconnect and leakage power are significant for nanometer FPGAs. [4] analyzes the leakage power of a commercial FPGA architecture in 90nm technology and quantifies the leakage power challenge. FPGA power optimization involves CAD algorithms and novel circuits and architectures. [5] proposes a configuration inversion method to reduce leakage power of multiplexers without additional hardware cost. [7] studies power-gating to reduce leakage power of unused FPGA logic blocks. [8, 9, 10] propose dual-Vdd and Vdd-programmable FPGA fabrics to reduce both dynamic and leakage power. [6] studies a suite of power-aware FPGA CAD algorithms without changing the existing FPGA circuits and architectures.

Conventional FPGA architecture has been evaluated using metrics of area and delay [17, 18], and recently power [3]. However, the emerging power-efficient circuits and architectures lead to different FPGA power characteristics, and therefore call for an architecture evaluation considering these power optimization techniques. In this paper, we study Vdd-programmable FPGAs which are originally proposed in [8, 9, 10]. We first improve the power model in an existing FPGA power evaluation framework to achieve high accuracy and fidelity. We then design a set of new Vdd-programmable circuits and develop several new architecture classes for Vdd-programmable FPGAs, which dramatically reduce the area overhead for Vdd programmability. Finally, we study the effect of cluster and LUT sizes on FPGA energy and delay, and evaluate the power saving by our new Vdd programmable architecture classes compared to single Vdd FPGAs.

The rest of the paper is organized as follows. Section 2 first describes FPGA architecture background, evaluation methodology, improved power model, and presents evaluation results for the baseline architecture class. Section 3 presents the novel circuit designs for Vdd-programmable and Vdd-gateable interconnect switches with reduced number of configuration SRAM cells. Sections 4 and 5 propose three Vdd-programmable architecture classes and evaluate their energy, delay and area with comparison to the baseline case. We conclude this paper in Section 6.

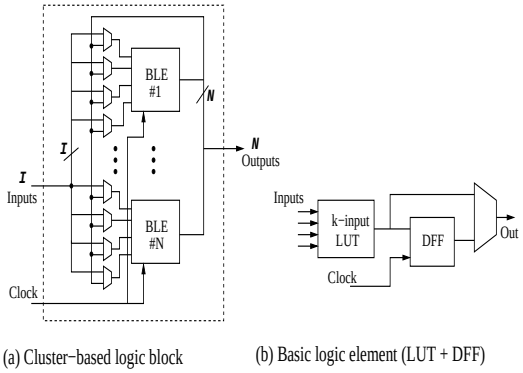
*This paper is partially supported by NSF CAREER award CCR-0093273, NSF grant CCR-0306682. Address comments to lhe@ee.ucla.edu.

FPGA'05, February 20–22, 2005, Monterey, California, USA.

2. BACKGROUND

2.1 Cluster-based Island Style FPGAs

We assume cluster-based island style FPGA architecture such as that in [11, 3] for all classes of FPGAs studied in this paper. Figure 1 shows the cluster-based logic block, which includes N fully connected Basic Logic Elements (BLEs). Each BLE includes one k-input lookup table (LUT) and one flip-flop (DFF). The combination of cluster size N and LUT size k are the architectural parameters we evaluate in this paper. The routing structure is of the island style shown in Figure 2. The logic blocks are surrounded by routing channels consisting of wire segments. The input and output pins of a logic block can be connected to the wire segments in the channels via a *connection block* (see Figure 2 (b)). A routing *switch block* is located at the intersection of a horizontal channel and a vertical channel. Figure 2 (c) shows a subset switch block [12], where the incoming track can be connected to the outgoing tracks with the same track number[1]. The connections in a switch block (represented by the dashed lines in Figure 2 (c)) are programmable routing switches. We implement routing switches by tri-state buffers and use two tri-state buffers for each connection so that it can be programmed independently for either direction. We decide the routing channel width W in the same way as the architecture study in [11], i.e., $W = 1.2W_{min}$ and W_{min} is the minimum channel width required to route the given circuit successfully.



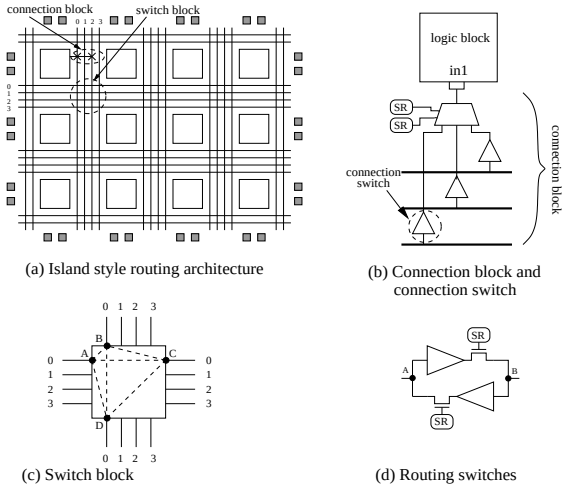
(a) Cluster−based logic block (b) Basic logic element (LUT + DFF)

Figure 1: FPGA logic block and basic logic element.

2.2 Evaluation Framework and Improved Power Model

Two FPGA power evaluation frameworks have been proposed recently [2] [3]. This paper uses fpgaEVA-LP [3] as the evaluation framework but improves its power model. fpgaEVA-LP includes a *BC-netlist* generator and a cycle-accurate power simulator. The BC-netlist generator takes the VPR placement and routing result and generates the Basic Circuit netlist (BC-netlist) annotated with post-layout capacitances and delay. The power simulator then performs cycle-accurate simulation on the BC-netlist to obtain FPGA power consumption. The mixed-level power model in the power simulator applies switch-level model to interconnects and macro-model to Lookup Tables (LUTs). We use five circuits from the MCNC benchmark set to illustrate the accuracy and fidelity of fpgaEVA-LP compared to SPICE simulation. The five circuits are chosen so that the circuit size is within the capability of SPICE simulation. They are mapped into 4-LUTs and packed into clusters with a

[1]Without loss of generality, we assume subset switch block in this paper.



(a) Island style routing architecture (b) Connection block and connection switch

(c) Switch block (d) Routing switches

Figure 2: (a) Island style routing architecture; (b) Connection block; (c) Switch block; (d) Routing switches.

cluster size of four. The largest circuit occupies six clusters and the smallest circuit occupies two clusters. As shown by the comparison in Figure 3, fpgaEVA-LP with the original power model achieves high fidelity but consistently underestimates total FPGA power.

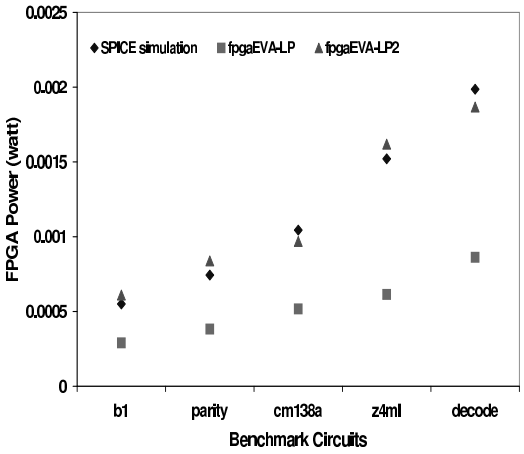


Figure 3: Comparison between SPICE simulation and cycle-accurate power simulation. The new power model significantly improves accuracy and still achieves high fidelity.

We improve the power model from [3] and call the new power evaluation framework with the improved power model as fpgaEVA-LP2. First, both [3] and [2] assumes that the short-circuit power of a routing buffer is a constant percentage of buffer switching power. In reality, a different signal transition time leads to different short-circuit power. [3] has shown that the short-circuit power is proportional to the signal transition time for a given load capacitance, however, it uses an average transition time and loses accuracy. In this paper, we store the relation between the percentage of short-circuit power in buffer switching power and the signal transition time (under the Elmore delay model), and calculate the short-circuit power value according to the signal transition time. Second, [3] assumes that the output signal transition time is twice

of the buffer delay. This simplistic assumption was originally used in gate sizing [13] and it is valid when the input signal is a step function and the output signal is a ramp function. We use SPICE to simulate a typical routing path in an FPGA, where a routing switch drives a wire segment and other routing switches. The input signal is no longer a step function because it is from the output of a routing switch in the previous stage. The output signal under a large load capacitance, which is usually the case in FPGAs, is not a perfect ramp function and the 10%-90% transition time for the output signal can be signifi cantly larger than twice of the buffer delay. In theory, the output signal transition time t_r can be expressed as $t_r = \alpha * t_{buffer}$ with t_{buffer} as the buffer delay. We use SPICE simulations to determine the parameter α for different buffer delays, which covers the cases of various input signal transition time and different load capacitance. Table 1 presents the values of α decided by our experiments and used in fpgaEVA-LP2. Finally, our FPGA circuits only apply gate-boosting [11] to routing switches in the channels. The output of multiplexers in logic clusters can have a voltage level degradation and the local buffers at the multiplexer output will have larger leakage power. We modify the power model in fpgaEVA-LP so that it can also consider local multiplexers without gate-boosting. We compare SPICE simulation with fpgaEVA-LP2 in Figure 3. Clearly, we still achieve high fi delity but improve the accuracy signifi cantly. The average of absolute error is 8.26% for the fi ve test circuits.

buffer delay	$< 0.012ns$	$< 0.03ns$	$>= 0.03ns$
α	2	4.4	7

Table 1: The Value of Parameter α to determine signal transition time.

In this paper, we use Berkeley predictive device model [14] and ITRS predictive interconnect model [15] for semi-global interconnects at the 100nm technology node. Table 2 summarizes the values of some key model parameters. The device and interconnect models are used throughout the rest of the paper.

Device model			
	Vdd (V)	NMOS-Vt (V)	PMOS-Vt (V)
normal-Vt	1.3	0.2607	-0.3030
high-Vt	1.3	0.4693	-0.5454
Interconnect model			
wire width	wire spacing	wire thickness	dielectric const.
0.56um	0.52um	1.08um	2.7

Table 2: Device and interconnect model at 100nm technology.

2.3 Evaluation Methodology and Results for Baseline Architecture Class

Our architecture evaluation methodology starts with VPR placement and routing results. For a given FPGA architecture and benchmark circuit, VPR can generate different placement and routing results by using different seeds in its placement algorithm. Figure 4 shows the FPGA energy and delay using ten different VPR seeds for the same circuit s38584. We label the seed value beside each data point. The delay variation is 12% and the energy variation is 5%. This variation due to VPR seeds may affect our architecture evaluation. Because the delay variation is more sensitive to the VPR seeds than the energy variation, we decide to use the min-delay solution among all VPR seeds for every benchmark circuit. Note that the min-delay solution often consumes low energy too. For the architecture evaluation in this paper, Energy (E), Delay (D) and Energy-Delay Product (ED) are always the geometric means of those values over 20 MCNC benchmark circuits.

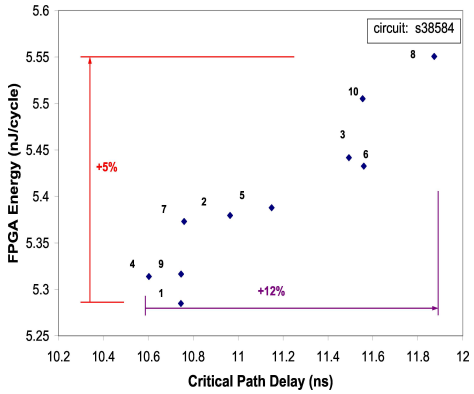


Figure 4: Impact of random seed on FPGA energy and delay.

Using the above methodology, we perform an architecture evaluation for the single-Vdd dual-Vt FPGA architectures from [8], also called FPGA *Class0* in this paper. The entire FPGA uses the uniform supply voltage 1.3V, but high-Vt is applied to all the FPGA confi guration SRAM cells to reduce SRAM leakage power. The high-Vt confi guration cells do not incur runtime performance degradation because they are constantly in read status after an FPGA is confi gured, and their read and write are irrelevant to the runtime performance. This high-Vt SRAM technique has already been used in commercial FPGAs [16].

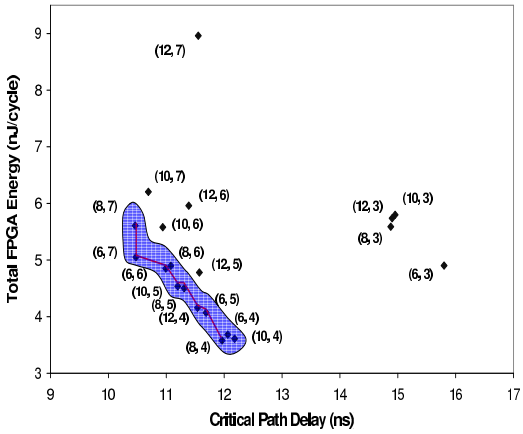


Figure 5: Energy-delay tradeoff for single-Vdd dual-Vt FPGA class (Class0). The polyline represents the strictly dominant architectures and the enclosed area covers the relaxed dominant architectures.

Figure 5 presents the evaluation results for FPGA Class0. Each data point in the fi gure is an FPGA architecture represented by a tuple (N, k), where N is the cluster size and k is the LUT size. If one architecture (N_1, k_1) has smaller delay and less energy consumption than another architecture (N_2, k_2), we say that architecture (N_1, k_1) is superior to (N_2, k_2). We defi ne *strictly energy-delay dominant architectures* as the set of superior data points in the entire energy-delay tradeoff space. Those architectures are highlighted by the polyline in Figure 5. Our results also show that

some of the architectures may have fairly similar energy and delay such as architectures $(N = 8, k = 4)$, $(N = 6, k = 4)$ and $(N = 10, k = 4)$, and all of of them can be valid solutions in reality. To avoid pruning out architectures with similar energy and delay, we further define *relaxed energy-delay dominant architectures*. If architectures (N_1, k_1) and (N_2, k_2) have both energy and delay difference less than $r\%$ (*relaxation parameter*), then neither of them can dominate the other one. With $r = 2$ in this paper, the relaxed dominant architectures are data points inside the enclosed curve in Figure 5. Min-delay and min-energy architectures are the two extreme cases among those energy-delay dominant architectures. The min-delay architecture is $(N = 8, k = 7)$ and the min-energy architecture is $(N = 8, k = 4)$ for the FPGA Class0 in Figure 5, and the energy and delay differences between the two extreme cases are 57% and 14%, respectively. It shows that a significant tradeoff between energy and delay can be obtained by varying cluster size and LUT size. Note that our min-energy architecture $(N = 8, k = 4)$ is also the min-area architecture found by [18]. Commercial FPGAs such as Xilinx Virtex-II [19] coincidently use a cluster size of 8 and an LUT size of 4, and therefore their architectures may have used min-area solution and turn out to be a min-energy architecture in single-Vdd architecture class.

3. FPGA CIRCUITS FOR VDD PROGRAMMABILITY

3.1 Previous Work and Overview

Vdd programmability has been introduced in [8, 9] and applied to logic blocks to reduce FPGA power. In this paper, Vdd programmability is defined as the flexibility to select Vdd levels for one used circuit element and the capability to power-gate an unused circuit element. Figure 6 shows the Vdd-programmable logic block. Two extra PMOS transistors, called *power switches*, are inserted between the conventional logic block and the dual-Vdd power rails for Vdd selection and power-gating.

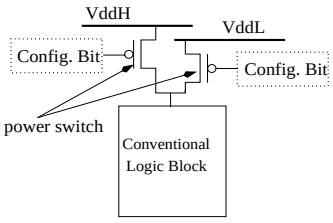


Figure 6: Vdd-programmable logic block.

[10] further extends programmable dual-Vdd to interconnect switches. Figure 7 shows the original design of Vdd-programmable interconnect switches (both routing switch and connection switch) in [10]. A level converter is needed whenever a low Vdd (VddL) interconnect switch drives a high Vdd (VddH) interconnect switch. In other cases, the level converter can be bypassed. As shown in Figure 7 (a), a pass transistor M1 and a MUX together with a configuration SRAM cell can be used to implement a configurable level conversion. For Vdd-programmable routing switch as shown in Figure 7 (b), two PMOS power switches M3 and M4 are inserted between the tri-state buffer and VddH, VddL power rails, respectively. Turning off one of the power switches can select a Vdd level for the routing switch. By turning off both power switches, an unused routing switch can be power-gated. SPICE simulation shows that power-gating the routing switch can reduce leakage power by a factor of over 300. As power switches stay either ON or OFF after configuration and there is no charging and discharging at their source/drain capacitors, the dynamic power overhead is almost negligible. The delay overhead associated with the power switch insertion can be bounded by around 6% when the power switch is properly sized. Another type of routing resources is the connection block [11] as shown in Figure 7 (c). The multiplexer-based implementation chooses only one track in the routing channel and connects it to the logic block input pin. The buffers between the routing track and the multiplexer are connection switches. Similar to the routing switch, programmable-Vdd is also applied to the connection switch.

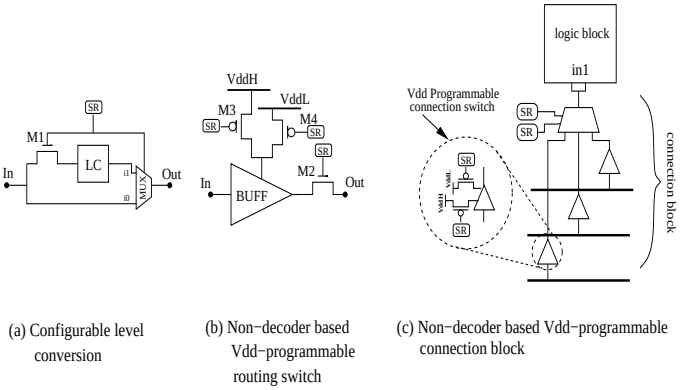


Figure 7: (a) Configurable level conversion; (b) Vdd-programmable routing switch; (c) Vdd-programmable connection block. (SR stands for SRAM cell and LC stands for level converter.)

However, a large number of configuration SRAM cells are introduced to provide Vdd programmability for interconnect switches in [10]. In this paper, we design two new types of interconnect switches, Vdd-programmable switch and Vdd-gateable switch, with reduced number of configuration SRAM cells and call them SRAM-efficient circuits. Similar to that in [10], a SRAM-efficient Vdd-programmable switch provides three states which are VddH, VddL and power-gating. In contrast, our SRAM-efficient design reduces the number of extra SRAM cells for Vdd programmability, therefore reduces SRAM leakage. Different from a Vdd-programmable switch, a Vdd gateable switch only has two states, to be programmed as being supplied with a pre-determined Vdd level or being power-gated when unused, but it can dramatically reduce the number of SRAM cells for Vdd programmability. The detailed circuit designs of SRAM-efficient Vdd-programmable and Vdd-gateable switches are discussed in the following sections.

3.2 SRAM Efficient Vdd-programmable Interconnect Switch

The Vdd-programmable interconnect switch [10] introduces a large number of extra configuration SRAM cells. As shown in Figure 7, there are three SRAM cells for each Vdd-programmable routing switch. 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 multiplexer and the other $2N$ extra SRAM cells are for N Vdd-programmable connection switches. We can use combinational logic such as decoder to reduce the number of extra SRAM cells introduced by Vdd programmability. Figure 8 shows the SRAM-efficient design of Vdd-programmable interconnect switches. As shown in Figure 8 (a), We first define a Vdd-programmable switch module with three signal

ports, $VddH\_En$, $VddL\_En$ and $Pass\_En$. By setting these three control signals, we can program Vdd-programmable switch between Vdd selection and power-gating. We design SRAM-efficient Vdd-programmable routing switch in Figure 8 (b). $Pass\_En$ can be generated by $VddH\_En$ and $VddL\_En$ with a NAND2 gate. Table 3 summarizes the configurations and relevant control signals for Vdd-programmable routing switch.

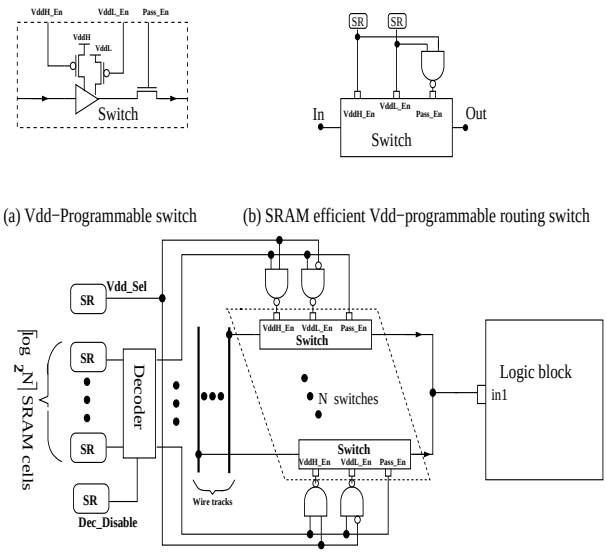


Figure 8: (a) Vdd-programmable switch (b) SRAM-efficient Vdd-programmable routing switch; (c) SRAM-efficient Vdd-programmable connection block.

state	$VddH\_En$	$VddL\_En$	$Pass\_En$
VddH	0	1	1
VddL	1	0	1
power-gated	1	1	0

Table 3: Configurations for a Vdd-programmable routing switch.

Similarly, Figure 8 (c) shows the SRAM-efficient design of Vdd-programmable connection block. For a connection block containing N connection switches, we use a $\lceil log_2 N \rceil : N$ decoder and $2N$ NAND2 gates as the control logic. There is a disable signal $Dec\_Disable$ for decoder. Each decoder output is connected to $Pass\_En$ of one connection switch. Setting $Pass\_En$ of a connection switch to '0' can power-gate this switch by setting both $VddH\_En$ and $VddL\_En$ to '1' with NAND2 gates. When the whole connection block is not used, all N outputs of the decoder are set to '0' to power-gate all the connection switches by asserting $Dec\_Disable$. When the connection block is in use, $Dec\_Disable$ is not asserted. By using $\lceil log_2 N \rceil$ configuration bits for the decoder, only one $Pass\_En$ is set to '1' and others are set to '0', i.e., only one connection switch inside the connection block is selected and connects the one track to logic block input, and other unused connection switches are power-gated. Another configuration bit Vdd_Sel is used to select the Vdd level for the selected connection switch. Table 4 summarizes the truth table of configurations and relevant control signals for Vdd-programmable connection switch.

state	$Dec\_Disable$	Vdd_Sel	$Pass\_En$	$VddH\_En$	$VddL\_En$
power-gated	1	-	0	-	-
power-gated	0	-	0	-	-
VddH	0	1	1	0	1
VddL	0	0	1	1	0

Table 4: Configurations for a Vdd-programmable connection switch.

For a connection block containing N connection switches, only $\lceil log_2 N \rceil + 2$ configuration SRAM cells are needed to provide Vdd selection and power-gating capability for each individual connection switch inside the connection block. Compared to a conventional connection block, only two extra configuration SRAM cells are introduced for Vdd selection and power-gating. Similar to the SRAM cell, we use high-Vt transistors for control logic to reduce leakage overhead as the delay of control logic will not affect system runtime performance. Similarly, we use minimum width transistors for control logic to reduce area overhead. Table 5 shows the comparison of the number of configuration SRAM cells, leakage and area between the original designs of Vdd-programmable routing switch/connection block in Figure 7 and our SRAM-efficient designs in Figure 8. As shown in Table 5, we can see that the SRAM-efficient designs of Vdd-programmable routing switch and connection block give us smaller area and less leakage. In the rest part of the paper, we only consider SRAM-efficient deign for Vdd-programmable interconnect switches.

3.3 Vdd-gateable Interconnect Switch

Compared to Vdd-programmable switch, Vdd-gateable interconnect switch only provides two states between a pre-determined Vdd level and power-gating, but it can dramatically reduce the number of extra SRAM cells for Vdd programmability. Figure 9 (a) shows the circuit design for a Vdd-gateable switch. Based on a conventional tri-state buffer, we insert a PMOS transistor M2 between the power rail and the tri-state buffer to provide the power-gating capability. When a switch is not used, transistor M1 is turned off by the configuration cell SR. At the same time, we can turn off M2 to perform power-gating for the unused switch. Similarly, both M1 and M2 are turned on by the configuration cell SR when the switch is used. Thus, we do not need to introduce an extra SRAM cell for power-gating capability. Figure 9 (b) presents Vdd-gateable routing switches. We can achieve leakage power reduction by a factor of over 300 for an unused switch when it is power-gated. However, there is a delay overhead associated with the M2 insertion. The Vdd-gateable routing switch suffers from delay increase compared to the conventional switch because M2 is inserted in series. We properly size M2 for the tri-state buffer to achieve a delay increase bounded by 6%. Similar to Vdd-programmable switch, dynamic power overhead associated with the insertion of PMOS M2 is almost negligible because transistor M2 is always ON when the routing switch is used and there is no charging or discharging occur at its source/drain capacitors.

The design of Vdd-gateable connection block is shown in Figure 9 (c). We only need $\lceil log_2 N \rceil$ configuration SRAM cells to control N connection switches in a connection block via a decoder and achieve the power-gating capability for each connection switch at the same time. We use another configuration bit, $Dec\_Disable$, to disable the decoder when we apply power-gating to the whole connection block. Similar to the SRAM-efficient design of Vdd-programmable switch, we use high-Vt and minimum width transistor for the decoder to reduce leakage and area overhead. Alternatively, N configuration SRAM cells can be used to control the same number of connection switches without using the decoder. Table 6 shows the comparison of the number of SRAM cells, leakage and area for a non-decoder

Vdd-programmable routing switch [10]			SRAM-efficient Vdd-programmable routing switch					compared to baseline [10]		
SRAM cells			SRAM cells			NAND2				
number	leakage (watt)	area	number	leakage (watt)	area	leakage (watt)	area	Δ number of SRAM cells	Δ leakage (watt)	Δ area
3	2.32E-8	21.87	2	1.55E-8	14.58	3.49E-10	2.50	-1	-7.38E-9	-4.79
32:1 connection block										
Vdd-programmable connection block [10]			SRAM-efficient Vdd-programmable connection block					compared to baseline [10]		
SRAM cells			SRAM cells			control logic				
number	leakage (watt)	area	number	leakage (watt)	area	leakage	area (watt)	Δ number of SRAM cells	Δ leakage (watt)	Δ area
69	5.32E-7	503.01	7	5.42E-8	43.74	3.30E-8	311	-62	-4.56-E7	-148.27

Table 5: The comparison of the number of SRAM cells, leakage and area between a Vdd-programmable routing switch/connection block and SRAM-efficient Vdd-programmable routing switch/connection block. We use 32:1 connection block and the control logic for SRAM-efficient design contains a standard 5:32 decoder and 64 NAND2 gates. Area is presented in terms of the number of minimum width transistor area.

Comparison between non-decoder based and decoder based 32:1 Vdd-gateable connection block										
non-decoder based connection block			decoder based connection block					comparison baseline: w/o decoder		
SRAM cells			SRAM cells			5:32 decoder				
number	leakage (watt)	area	number	leakage (watt)	area	leakage (watt)	area	Δ number of SRAM cells	Δ leakage (watt)	Δ area
32	2.47E-7	233.28	6	4.63E-8	43.74	2.00E-8	94.25	-26	-1.81E-7	-95.29

Table 6: The comparison of the number of SRAM cells, leakage and area between a non-decoder based Vdd-gateable connection block and a decoder based Vdd-gateable connection block. We use a 32:1 connection block. For the decoder based Vdd-gateable connection block, we use a 5:32 decoder with complementary output. Area is presented in terms of the number of minimum width transistor area.

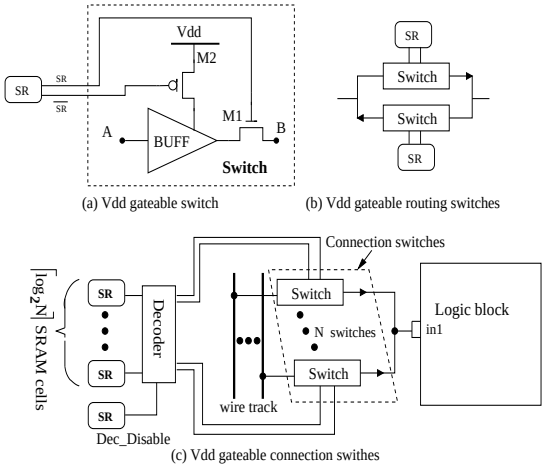


Figure 9: (a) Vdd-gateable switch; (b) Vdd-gateable routing switches; (c) Vdd-gateable connection switches. (SR stands for SRAM cell)

based and decoder based connection block containing 32 connection switches. As shown in Table 6, the decoder based Vdd-gateable connection block consumes less area and leakage power compared to the non-decoder based design. In the rest part of this paper, we only consider decoder based Vdd-gateable connection block that only introduces one extra configuration SRAM cell.

4. ARCHITECTURE EVALUATION FOR VDD PROGRAMMABLE FPGAS

In this section, we first evaluate two architecture classes *Class1* and *Class2* for Vdd programmable FPGAs. Class1 applies programmable dual-Vdd to all the logic blocks and routing switches, and inserts a configurable level conversion circuit in front of each routing switch as well as at the inputs/outputs of the logic blocks. It is the same fully Vdd-programmable architecture style proposed in [10], but we use the SRAM-efficient circuit design for routing and connection switches to reduce the number of SRAM cells for Vdd programmability. Class2 applies programmable dual-Vdd only to logic blocks, and uses Vdd-gateable routing/connection switches in FPGA interconnects. Therefore, the interconnect switches in architecture Class2 only have two configurable states: high Vdd (VddH) and power-gating. As we use VddH for interconnects in architecture Class2, level converters are only needed at the logic block outputs, but not at the logic block inputs nor in the routing channels. Similar to the baseline architecture Class0 in 2.3, the configuration SRAM cells in both architecture classes use the high Vt SRAM design. All these architecture classes (with Class3 to be presented in Section 5) are summarized in Table 7.

Architecture Class	Logic block	Interconnect
class0 (baseline)	single Vdd	single Vdd
class1	programmable dual-Vdd	programmable dual-Vdd w/ LCs in the routing
class2	programmable dual-Vdd	Vdd-gateable
class3	programmable dual-Vdd	programmable dual-Vdd w/o LCs in routing chann.

Table 7: Summary of baseline architecture class and Vdd-programmable architecture classes (LC denotes the level converter).

In our architecture evaluation framework, we use a simple design flow similar to that in [9, 10]. Starting with a single-Vdd gate level netlist, we apply technology mapping and timing-driven packing [11] to obtain the single-Vdd cluster-level netlist. We then perform single-Vdd timing-driven placement and routing by VPR [11] and generate the basic circuit netlist (BC-netlist). We assume that the initial Vdd level is VddH everywhere, and calculate power sensitivity $\Delta P/\Delta V_{dd}$, which is the power reduction by changing VddH to VddL for each circuit element. The total power P includes both switching power P_{sw} and leakage power P_{lkg}. A greedy algorithm is carried out for Vdd assignment considering iteratively updated timing slack (See Figure 10). Note that Vdd assignment is performed after single-Vdd (VddH) routing, and the placement and

routing solution is the same in all FPGA classes for each benchmark circuit. For FPGA Class1, the Vdd assignment unit is a logic block or an interconnect switch. For FPGA Class2, the Vdd assignment unit is a logic block. For both Class1 and Class2, power-gating is applied to all unused logic blocks and programmable switches. Finally, we perform the energy and delay evaluation for the dual-Vdd design.

Sensitivity-based dual-Vdd assignment algorithm:
Assign VddH to all assignment units;
Calculate power-sensitivity S for all assignment units;
While($\exists$ assignment units not tried)
{
  Assign VddL to the unit with largest S if no critical path increase;
  Update timing slack and mark the unit as tried;
}

Figure 10: Sensitivity-based dual-Vdd assignment algorithm.

Figure 11 presents the energy-delay tradeoff in terms of different architectures, i.e., different combinations of cluster size N and LUT size k, for three FPGA classes: Class0, Class1 and Class2. Considering that the optimal VddL/VddH ratio is $0.6 \sim 0.7$ as studied in [20], we use 1.3v for high Vdd and 0.8v for low Vdd in our experiments. We only show the relaxed dominant architectures in the fi gure and the polyline represents the strictly dominant architectures. Similar to the baseline FPGA class0, the min-delay architecture is $(N = 8, k = 7)$ for both class1 and class2. The min-energy architecture is $(N = 8, k = 4)$ for class1 and $(N = 12, k = 4)$ for class2. This shows that LUT size 7 gives the best performance and LUT size 4 leads to the lowest energy consumption for these Vdd programmable FPGAs.

We then use the metrics of energy E, delay D and energy-delay product ED to compare the two classes of Vdd-programmable FPGAs (Class1 and Class2) and the baseline FPGA (Class0). We use the min-energy (min-delay) architecture within each FPGA architecture class and obtain the energy saving (delay increase) by Vdd programmable FPGAs. Compared to baseline architecture class, FPGA Class1 obtains an energy saving of 22.63% and FPGA Class2 obtains an energy saving of 44.42%. The delay increase due to Vdd programmability is only 4% for both FPGA Class1 and Class2. As the routing solution is same for each benchmark in single-Vdd FPGA Class0 and Vdd-programmable FPGA Class1, Class2, this system level performance degradation reflects the impact of delay increase due to Vdd-programmability in circuit level. We also use the min-ED (i.e., the minimum energy-delay product) architecture within each architecture class and obtain the ED product reduction. FPGA Class1 reduces ED product by 19.48% and Class2 reduces ED product by 44.14%.

Vdd programmability increases the total number of SRAM cells required to store those extra confi guration bits. However, SRAM cells are vulnerable to soft errors and the total number of SRAM cells should be minimized. Table 9 presents the increase in SRAM cell number and the total device area overhead due to Vdd programmability. The SRAM cells include those used in LUTs and the total device area includes both logic block and interconnect device area. Only dominant architectures are shown in the table. Vdd programmable FPGA Class1 increases the SRAM cell number by 132%. This shows that fully Vdd programmable FPGAs need a large number of extra SRAM cells to provide Vdd programmability. FPGA Class2 only increases SRAM cell number by 3% because only two states (VddH and power-gating) are provided for FPGA interconnect switches and the original SRAM cells for interconnection programmability can be shared for Vdd programmability.

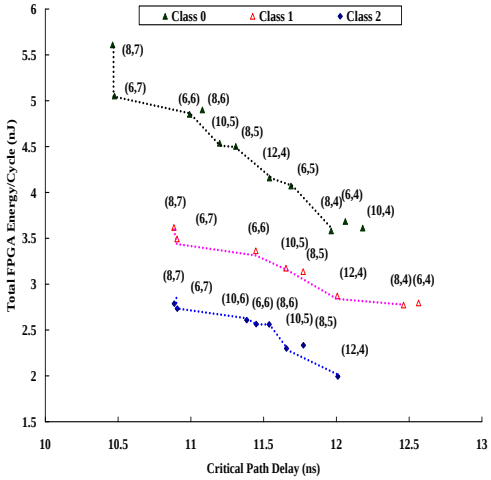


Figure 11: Energy and delay tradeoff for the baseline single-Vdd dual-Vt FPGA (Class0) and the two classes of Vdd-programmable FPGAs (Class1 and Class2). The figure only shows relaxed energy-delay dominant solutions and the strictly dominant solutions are represented by polylines.

Considering total device area overhead including the extra SRAM cells, power switches and level converters, FPGA Class1 has 178% area overhead and FPGA Class2 has 48% area overhead.

5. IMPROVED FPGA ARCHITECTURES

5.1 FPGA Architectures and Related CAD Algorithm

By using Vdd-programmable interconnects, we can reduce the interconnect dynamic energy which is not available by Vdd-gateable interconnects. However, as presented in Section 4, FPGA fully Vdd-programmable architecture Class1 consumes more energy than FPGA architecture Class2 which uses Vdd-gateable interconnects. This is because of the leakage overhead of the large number of level converters in routing channels, which provides Vdd programmability for each individual interconnect switch. To achieve

Arch. Class →	Class0 (baseline)	Class1	Class2	Class3
min-E arch. (N,k)	(8,4)	(8,4)	(12,4)	(12,4)
energy (nJ/cycle)	3.58	2.77	1.99	1.81
energy saving (%)	-	22.63%	44.42%	49.41%
min-D arch. (N,k)	(8,7)	(8,7)	(8,7)	(8,7)
delay (ns)	10.46	10.88	10.89	10.88
delay increase (%)	-	4%	4%	4%
min-ED arch. (N,k)	(8,4)	(12,4)	(12,4)	(12,4)
ED product (nJ · ns)	42.82	34.48	23.92	21.75
ED reduction	-	19.48%	44.14%	49.21%
device area	7014240	27245631	13943892	21557990
area overhead	-	288%	99%	207%

Table 8: Comparison between Vdd programmable FPGAs (Class1 and Class2) and the baseline FPGA (Class0) using energy E, delay D, energy-delay product (ED) and device area in minimum width transistor area.

Dominant Arch. (N,k)	total # of SRAM cells on chip			total device area			total # of SRAM cells	total device area
	Class0 baseline	Class1 (% overhead)	Class2 (% overhead)	Class0 baseline	Class1 (% overhead)	Class2 (% overhead)	Class3 (% overhead)	Class3 (% overhead)
(8,7)	649218	88%	2%	11541440	149%	41%	17%	102%
(6,7)	621929	89%	2%	10689783	164%	42%	20%	116%
(6,6)	469504	128%	3%	10114162	195%	56%	31%	140%
(10,5)	374174	164%	3.4%	9793576	189%	50%	33%	129%
(12,4)	317391	190%	4%	9173613	197%	52%	40%	135%
Average	-	132%	3%	-	178%	48%	28%	124%

Table 9: Total number of configuration SRAM cells and device area overhead for different Vdd programmable FPGAs. SRAM cells include those used in LUTs and total device area includes both logic block and interconnect area. The device area is in minimum width transistor area.

better energy-delay tradeoff, we design an improved fully Vdd-programmable FPGA architecture *Class3*. It uses the same SRAM-efficient interconnect switches as FPGA architecture Class1, but inserts level converters only at logic block inputs and outputs. Since there is no level converter in routing channels, we need a CAD algorithm to guarantee that no VddL interconnect switch drives VddH interconnect switch. We tackle the problem by choosing routing tree as the Vdd assignment unit. Similar to FPGA Class1, the same design flow and the sensitivity-based Vdd level assignment algorithm in Figure 10 is used to decide the Vdd level for each routing tree. The only difference is that we use a routing tree as the assignment unit for FPGA Class3 while an interconnect switch is used as the assignment unit for Class1. Since two routing trees will not intersect with each other in routing channels, we do not need level converters in routing channels. Same as FPGA Class1, the Vdd assignment is performed after single-Vdd (VddH) routing and using a routing tree as the assignment unit does not impose any additional routing constraint. Figure 12 illustrates the situation that a VddH routing tree and VddL routing tree can share a same track without level converters in routing channels.

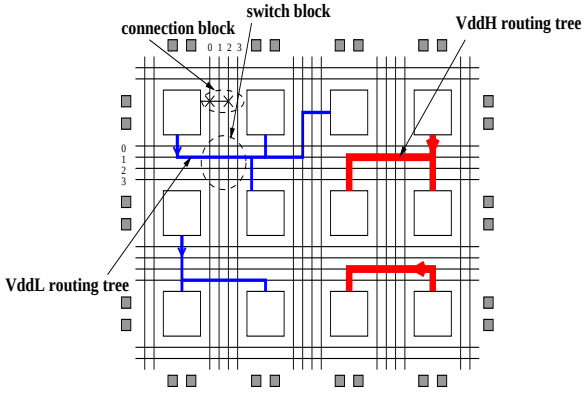


Figure 12: Improved fully Vdd-programmable FPGA architecture Class3. No level converter is inserted in routing tracks.

5.2 Evaluation Results and Discussions

In this section, we evaluate our improved fully Vdd-programmable architecture *Class3*. Figure 13 shows the energy-delay evaluation for our improved architecture Class3 compared to the evaluation results for architecture Class0, Class1 and Class2. As shown in Figure 13, we can see that the improved architecture Class3 can achieve better energy-delay tradeoff than architecture Class1, and even better than Class2. This is because FPGA Class3 removes the level converters in the routing channels, but still can reduce interconnect dynamic energy. This is not available in architecture Class2 which uses Vdd-gateable interconnect switches.

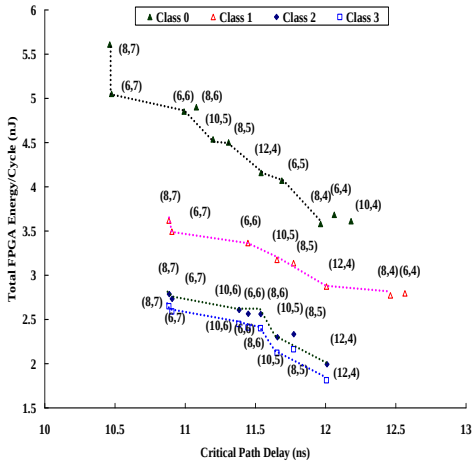


Figure 13: Energy and delay tradeoff for the baseline single-Vdd dual-Vt FPGA (Class0) and the three classes of Vdd-programmable FPGAs (Class1, Class2 and Class3). The figure only shows relaxed energy-delay dominant solutions and the strictly dominant solutions are represented by polylines.

Similar to Class0, the min-delay architecture is $(N = 8, k = 7)$ for Class3. The min-energy architecture is $(N = 12, k = 4)$ for Class3. $(N = 12, k = 4)$ also gives the minimum energy delay product ED in architecture Class3. We can see that for our improved FPGA architecture Class3, LUT size 7 always gives the best performance and LUT size 4 always leads to the lowest energy consumption. Compared to the min-energy (min-delay) architecture within baseline architecture Class0, the min-energy architecture in Class3 obtains an energy reduction of 49.41%, and the min-delay architecture in Class3 has a 4% delay overhead due to Vdd programmability. The min-ED architecture in FPGA Class3 reduces energy delay product ED by 49.21%. As shown in Table 8, FPGA Class3 gives the lowest energy as well as the lowest energy delay product ED. The total number of configuration SRAM cells and the total device area of FPGA Class3, compared with FPGA Class0, Class1 and Class2, are presented in Table 9. FPGA Class3 increases the number of configuration SRAM cells by 28.25% and the device area by 124% for Vdd programmability. Both Class2 and Class3 introduce smaller number of extra configuration SRAMs and give a smaller device area overhead while reducing more energy

compared to Class1. Compared to FPGA Class2, Class3 reduces more energy as well as ED while introducing more configuration SRAM cells and having a larger area overhead.

6. CONCLUSIONS AND DISCUSSIONS

We have developed an improved FPGA power model with high fidelity and accuracy, and designed novel Vdd-programmable and Vdd-gateable interconnect switches with significantly reduced number of configuration SRAM cells compared to [10]. Using the new switches, we have evaluated three new classes of Vdd-programmable FPGA architectures. *Class1* applies Vdd programmability to both logic blocks and interconnects, where Vdd-level converters are inserted before each interconnect switch. *Class2* uses Vdd-programmable logic blocks and Vdd-gateable interconnects. Similar to *Class1*, *Class3* applies Vdd programmability to both logic blocks and interconnects, but it does not insert any Vdd-level converter in routing channels. The baseline for comparison is *Class0*, which uses high-Vdd for both logic blocks and interconnects. High-Vt is applied to configuration SRAM cells for all four architecture classes, and the same dual-Vdd levels are applied to Class1$\sim$3. Using the metric of Energy-Delay Product (ED) measured as a geometric mean over the MCNC benchmark set, the ED reduction for the min-ED architecture in Class1, Class2 and Class3 is 19.48%, 44.14% and 49.21%, respectively. The SRAM cell overhead introduced by Vdd-programmability for Class1, Class2 and Class3 is 132%, 3% and 28%, respectively. The total device area overhead for Class1, Class2 and Class3 is 178%, 48% and 124%, respectively. Both FPGA Class2 and Class3 achieve more energy reduction with less SRAM and area overhead compared to FPGA Class1. Note that Class1 is the same as the Vdd-programmable FPGA architectures proposed in a very recent work [10] but with less power and area due to the new circuit designs. While FPGA Class3 gives the lowest energy consumption, FPGA Class2 achieves comparable energy reduction with significantly reduced number of SRAM cells and device area overhead. We conclude that Class2 is the best architecture class considering area, power and performance tradeoff. Our evaluation results also show that, within each architecture class, LUT size 4 gives the lowest energy consumption while LUT size 7 leads to the highest performance.

In this paper, area is represented by the number of minimum width transistor area. Our future work includes co-developing CAD algorithms and architectures to increase power reduction and decrease SRAM cell and area overhead due to Vdd programmability. Particularly, because high-Vdd wire segments may drive low-Vdd wire segments without using Vdd-level converters, we will apply dual-Vdd levels within a routing tree. This may reduce more dynamic power compared to using high-Vdd in the entire tree. Additionally, similar to a mix of high-Vdd, low-Vdd, and programmable-Vdd logic blocks used in [9], we will replace the programmable-Vdd interconnects in Class3 by an optimal mix of high-Vdd, low-Vdd and programmable-Vdd interconnects to reduce the area overhead with bounded or no performance loss.

Acknowledgement

The authors like to thank Mr. Lerong Cheng and Ms. Ho-Yan Phoebe Wong at UCLA for generating circuit models for a variety of basic FPGA circuits and for helpful discussions.

7. REFERENCES

[1] E. Kusse and J. Rabaey, "Low-energy embedded FPGA structures," in *Proc. Intl. Symp. Low Power Electronics and Design*, pp. 155–160, August 1998.

[2] 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.

[3] 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.

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

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

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

[7] 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.

[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*, February 2004.

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

[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.

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

[12] 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.

[13] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya, and S. M. Kang, "An exact solution to the transistor sizing problem for CMOS circuits using convex optimization," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, pp. 1621–1634, Nov. 1993.

[14] U. of Berkeley Device Group, "Predictive technology model," in *http://www.device.eecs.berkeley.edu/ ptm/mosfet.html*, 2002.

[15] International Technology Roadmap for Semiconductor in *http://public.itrs.net/*, 2002.

[16] S. Trimberger, "Private conversation," 2003.

[17] J. Rose et al, "Architecture of field-programmable gate arrays: The effect of logic functionality on area efficiency," *ISSCC*, 1990.

[18] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," in *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, pp. 3–12, Feb 2000.

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

[20] 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.*, pp. 495–498, 1998.