Device And Architecture Co-Optimization for FPGA Power Reduction

Lerong Cheng, Phoebe Wong, Fei Li, Yan Lin, Lei, He
Electrical Engineering Department
University of California, Los Angeles

Abstract

Device optimization considering supply voltage Vdd and threshold voltage Vt tuning does not increase chip area but has a great impact on power and performance in the nanometer technology. This paper studies the simultaneous evaluation of device and architecture optimization for FPGA. We first develop an efficient yet accurate timing and power evaluation method, called trace-based model. By collecting trace information from cycle-accurate simulation of placed and routed FPGA benchmark circuits and re-using the trace for different Vdd and Vt, we enable the device and architecture co-optimization for hundreds of combinations. Compared to the baseline FPGA which has the architecture same as the commercial FPGA used by Xilinx, and has Vdd suggested by ITRS but Vt optimized by our device optimization, architecture and device co-optimization can reduce energy-delay product by 20.5% and chip area by 20.1% compared to the conventional FPGA architecture. Furthermore, considering power-gating of unused logic blocks and interconnect switches, our co-optimization method reduces energy-delay product by 54.7% and chip area by 8.3%. To the best of our knowledge, this is the first in-depth study on architecture and device co-optimization for FPGAs.

I. Introduction

Field programmable gate array (FPGA) allows the same silicon implementation to be programmed or re-programmed for a variety of applications. It provides low NRE (non-recurring engineering) cost and short time to market. FPGA architecture has a significant impact on performance, area, and power. Earlier architecture evaluation has been conducted to study the performance and area impacts of lookup table (LUT) size K (number of inputs of an LUT) and cluster size N (number of LUTs per cluster) [1]–[3]. As technology continues scaling down to the nanometer feature size, (e.g., 100nm or below), power has become an important design constraint for FPGAs. Recent studies [4], [5] developed parameterized FPGA power models and evaluated power characteristics of existing FPGA architectures.

To reduce FPGA power, several circuits and architectures have been proposed, including region based power-gating of unused FPGA logic blocks [6], field programmability of Vdd for FPGA logic [7], [8] and interconnect [9]. Architecture evaluation considering Vdd-programmable FPGA has been conducted [10]. However, the supply voltage (Vdd) and threshold voltage (Vt) have great impact on power (especially leakage power) and delay in nanometer technologies. But all the aforementioned architecture evaluation assumed fixed Vdd and Vt [1]–[5], [10], and have not conducted simultaneous evaluation on device optimization such as Vdd and Vt tuning and architecture optimization on LUT and cluster size.

Vdd and Vt optimization has little or no area overhead compared to power gating and Vdd programmability. Architecture and device co-optimization is obviously able to give better power and performance tradeoff compared to architecture tuning alone. We define *hyper-architecture* (in short, hyper-arch) as the combination of device parameters and architectural parameters. The co-optimization requires the exploration of the following dimensions: cluster size N, LUT size K, supply voltage Vdd, and threshold voltage Vt. The total hyper-arch combinations can be easily over a few hundreds and calls for accurate yet extremely efficient timing and power evaluation methods.

The existing FPGA power evaluation methods are based on cycle-accurate simulation [4] or logic transition density estimation [5]. Timing and power are calculated for each circuit element. Therefore, it is very time-consuming to explore the huge hyper-arch solution space using methods from [4], [5]. The first contribution of this work is that we develop a trace-based estimator for FPGA power, delay, and area. We perform benchmark profiling and collect statistical information on switching activity, short circuit power, critical path structure, and circuit element utilization rate for a given set of benchmark circuits (MCNC benchmark set in this paper). We then derive formulae that use the statistical information and obtain FPGA performance and power for a given set of architectural and device parameter values. Our trace-based estimator has a high fidelity compared to the cycle-accurate simulation [4] and an average error of 3.4% for power and of 6.1% for delay. We will show that our trace information depends only on FPGA architecture but is *insensitive* to device parameters. Therefore, once the trace information is collected for the benchmark set, the remaining runtime is negligible as the trace-based hyper-arch evaluation is based on formulae and lookup tables. The trace collecting has the same runtime as evaluating FPGA architecture for a given Vdd and Vt combination using cycle accurate simulation. It took one week to collect the trace for the MCNC benchmark set using eight 1.2GHz Intel Xeon servers. But all the hyper-arch evaluation reported in this paper with over hundreds of Vdd and Vt combinations took a few minutes on one server.

The second contribution is that we perform the architecture and device co-optimization for a variety of FPGA classes. We explore different Vdd and Vt combinations in addition to the cluster size and LUT size combinations. For comparison, we obtain the baseline FPGA which uses the same architecture as the commercial FPGA used by Xilinx, and Vdd suggested by ITRS [11] but Vt optimized by our device optimization, and is significantly better than the one with no device optimization. Compared to the baseline FPGA, architecture and device co-optimization can reduce energy-delay product (product of energy per clock cycle and critical path delay, in short, ED) by 20.5 and chip area by 20.1%. Furthermore, considering power-efficient FPGA architecture with power-gating capability for logic blocks and interconnect switches, our architecture and device co-optimization method reduces ED by 54.7% and chip area by 8.3%.

The rest of the paper is organized as follows. Section II presents the background of FPGA architecture and introduces the cycle accurate power simulation. Section III presents our trace-based estimation models. Section IV applies the new estimation method and performs the architecture and device co-optimization. Section V concludes this paper.

II. Preliminaries

In this section, we will first give the introduction of FPGA architecture, then we will review the cycle-accurate simulation power model.

A. FPGA Architecture

We assume cluster-based island style FPGA architecture such as that in [4], [12] 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 is the architectural issue 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 routing 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 [13], 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 define an *interconnect segment* as a wire segment driven by a tri-state buffer or a buffer[2]. In this paper, we assumed all the wire segments spans 4 logic blocks, which is the best routing architecture for low power FPGA [7]. We decide the routing channel width CW in the same way as the architecture study in [12], i.e., $CW = 1.2CW_{min}$ where CW_{min} is the minimum channel width required to route the given circuit successfully. The channel width CW represents a "low-stress" routing situation that usually occurs in commercial FPGAs for 'average' circuits.

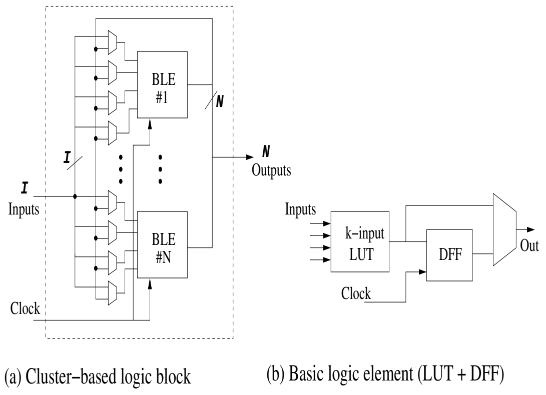
BLE #1
N
I
N Outputs
I Inputs
BLE #N
Clock
Inputs
k-input LUT
DFF
Out
Clock
(a) Cluster-based logic block
(b) Basic logic element (LUT + DFF)

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

Power gating can be applied to interconnects and logic blocks to reduce FPGA power. Figure 3 shows the circuit design of vdd-gateable interconnect switch and logic block. We insert a PMOS transistor between the power rail and the buffer (or logic block) to provide the power-gating capability. When a buffer or logic block is not used, gating transistor is turned off by the configuration cell. Spice simulation shows that power-gating can reduce the leakage power of an unused buffer or logic block by a factor of over 300. There are power and delay overhead associated with the power transistor insertion. The dynamic power overhead is almost negligible. This is because that the power transistor stay either ON or OFF after configuration and there is no charging and discharging at their source/drain capacitors. The delay overhead associated with the power transistor insertion can be bounded when the power transistor is properly sized. As shown in Section IV, we optimize the power transistor size to achieve best delay-area tradeoff. Notice that when power gating is applied, the input MUX for the connection box is no longer needed, as shown in Figure 3. This is because we can select one wire segment to connect to the logic block by turning

[1]Without loss of generality, we assume subset switch block in this paper.
[2]We interchangeably use the terms of switch and buffer/tri-state buffer.

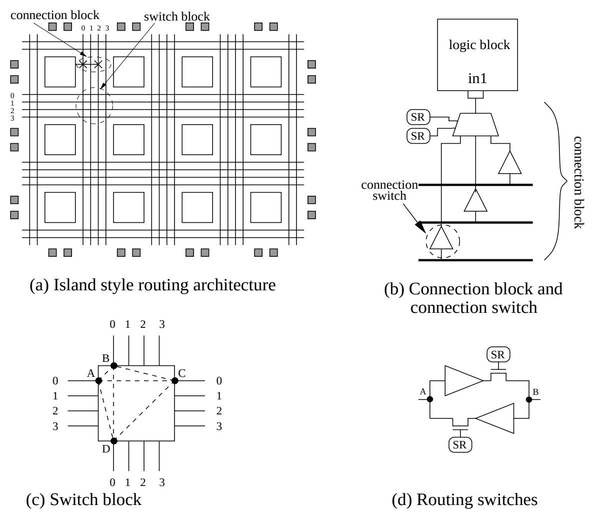

Fig. 2. (a) Island style routing architecture; (b) Connection block; (c) Switch block; (d) Routing switches.

one of the connection buffer on and power gating all the other connection buffers. Since we remove the imput MUX, we can significantly reduce the delay of the connection box. Therefore, applying power gating may even improve the performance in some cases. The detailed discussion of the impact of power gating will be in Section IV-C.

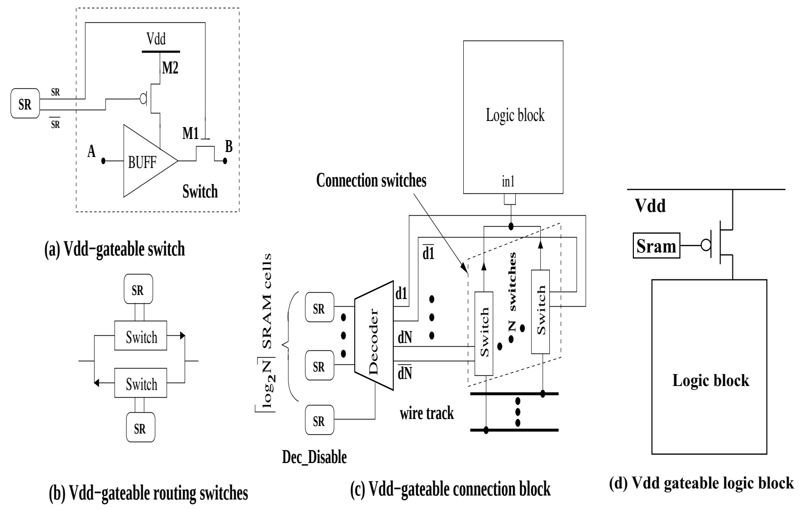

Fig. 3. (a) Vdd-gateable switch; (b) Vdd-gateable routing switch; (c) Vdd-gateable connection block; (d) Vdd-gateable logic block.

B. Cycle-Accurate Simulation

Given the above FPGA architecture, a detailed power model has been proposed for cycle-accurate simulation [4]. It models switching power, short-circuit power, and leakage power. The first two types of power are called *dynamic power* and they can only occur when a signal transition happens. The switching power is due to the charging and discharging of load capacitance, and can be modeled as follows,

$$P_{sw} = 0.5f \cdot V_{dd}^2 \cdot \sum_{i=1}^{n} C_i S_i \qquad (1)$$

where n is the total number of nodes, f is the clock frequency, Vdd is the supply voltage, C_i is the load capacitance for node i and S_i is the switching activity for node i. Short-circuit power occurs when there is a signal transition at a gate output and the pull-up and pull-down transistors conduct simultaneously for a short period of time. It is a function of signal transition time and load capacitance, and can be modeled as follows.

$$P_{sc} = P_{sw} \cdot \alpha_{sc}(t_r) \qquad (2)$$

where t_r is the signal transition time and $\alpha_{sc}(t_r)$ is the ratio between short-circuit power and switching power. α_{sc} depends on transition time t_r. The third type of power, *leakage power*, is consumed when there is no signal transition for a gate or a circuit module. It is a function of technology, temperature, static input vector, and stack effect of the gate type. average leakage power of a circuit element at given temperature, Vdd and Vt can be characterized by doing SPICE simulation under different input vectors. In each clock cycle of simulation, the simulation under real delay model obtains the number of signal transitions as well as transition time of a circuit element and calculate its dynamic power. If the circuit element has no signal transition in that cycle, it only consumes leakage power. Also, leakage power is consumed by an active element too. Essentially, the cycle-accurate simulation is needed to get the switching activity as well as signal transition time under real delay model.

III. Trace-based Estimation

The cycle-accurate simulation is very time consuming because a large number of the input vectors needs to be simulated using detailed delay model. Also, in order to obtain FPGA delay, static timing analysis has to be conducted for the entire circuit mapped to the FPGA fabric. The cycle-accurate simulation is not practical for architecture and device co-optimization because the total hype-arch combinations can be easily over a few hundreds. We develop a runtime efficient trace-based estimation method. For a given benchmark set and a given FPGA architecture, we collect statistical information of switching activity, critical path structure and circuit element utilization by profiling the benchmark circuits using cycle-accurate simulation. These statistical information is called the *trace* of the given benchmark set. We further develop our quick estimation formula based on trace information and circuit models at different technologies. We will show that the trace information is insensitive to the device parameters such as Vdd and Vt, and it can be reused during our device optimization to avoid the time-consuming cycle-accurate simulation. Figure 4 illustrates the relation between the cycle-accurate simulation and trace-based estimation. Table I summarizes the trace information we collect as well as the device and circuit parameters. In the table, trace parameters, including N_i^u, N_i^t, S_i^u, N_i^p and α_{sc}, are what only depend on FPGA architecture, device parameters, including Vdd and Vt, are what depend on technology scale, and circuit parameters, including P_i^s, C_i^u and D_i, are what depend on circuit design and device. The details of our trace-based models are discussed in the following.

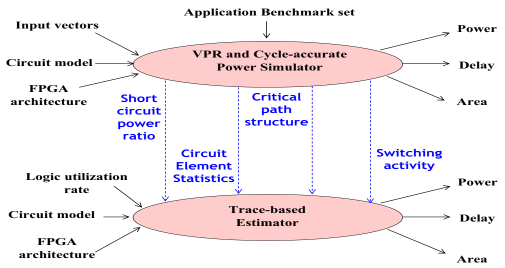

Fig. 4. Cycle-accurate simulation versus trace-based estimation.

Trace Parameters (depend on architecture)	
N_i^u	# of *used* circuit elements of resource type i
N_i^t	total # of circuit elements in resource type i
S_i^u	avg. switching activity for a *used* ckt element of type i
N_i^p	# of circuit elements of type i on the critical path
α_{sc}	ratio between short circuit power and switching power
Device Parameters (depend on technology)	
Vdd	power supply voltage
Vt	threshold voltage
Circuit Parameters (depend on circuit design and device)	
P_i^s	avg. leakage power for a circuit element in resource type i
C_i^u	avg. load capacitance of a circuit element of resource type i
D_i	avg. delay of a circuit element in resource type i

TABLE I
TRACE INFORMATION, DEVICE AND CIRCUIT PARAMETERS.

A. Dynamic Power Model

Dynamic power includes switching power and short-circuit power. A circuit implemented on a FPGA fabric cannot utilize all circuit elements in FPGA because of the programmability. Dynamic power is only consumed by the utilized FPGA resources. Our trace-based switching power model distinguishes different types of used FPGA resources and applies the following formula:

$$P_{sw} = \sum_i \frac{1}{2} N_i^u \cdot f \cdot V_{dd}^2 \cdot C_i^{sw} \tag{3}$$

The summation is over different types of circuit elements, i.e., LUTs, buffers, input pins and output pins. For circuit elements in FPGA resource type i, C_i^{sw} is the average switching capacitance and N_i^u is the number of used circuit elements, f is the operating frequency. In this paper, we assume the circuit works in its maximum frequency, i.e., the reciprocal of the critical path delay. The switching capacitance is further calculated as follows,

$$\begin{aligned} C_i^{sw} &= \left(\sum_{j \in El_i} C_{i,j}/N_i^u\right) \cdot S_i^u \\ &= C_i^u \cdot S_i^u \end{aligned} \tag{4}$$

For the type i circuit elements, C_i^u is the average load capacitance of a used circuit elements, which is average over $C_{i,j}$, the local load capacitance for used circuit element j, El_i is the set of used type i circuit elements, and S_i^u is the average switching activity of used type i circuit elements. We assume that the average switching activity of the circuit elements is determined by the circuit logic functionality and FPGA architecture. The device parameters of Vdd and Vt have a limited effect on switching activity. We verify this assumption in Table II by showing the average switching activity of five benchmarks at different Vdd and Vt levels.

benchmark	70nm Vdd=1.1 Vt=0.25		100nm Vdd=1.3 Vt=0.32		70nm Vdd=1.0 Vt=0.20	
	logic	interconnect	logic	interconnect	logic	interconnect
alu4	2.06	0.55	2.01	0.54	2.03	0.59
apex2	1.73	0.47	1.75	0.47	1.70	0.47
apex4	1.23	0.27	1.19	0.26	1.16	0.29
bigkey	1.75	0.56	1.96	0.59	1.71	0.55
clma	0.90	0.21	0.87	0.21	0.91	0.23

TABLE II

SWITCHING ACTIVITY COMPARISON FOR DIFFERENT TECHNOLOGY SCALE, VDD AND VT. ARCHITECTURE SETTING: $N = 10, K = 4$.

The short circuit power is related to signal transition time, which is difficult to obtain without detailed simulation under real delay model. In our trace-based model, we model the short circuit power as:

$$P_{sc} = P_{sw} \cdot \alpha_{sc} \tag{5}$$

Where α_{sc} is the ratio between short circuit power and switching power. Such ratio value is a circuit parameter depending on FPGA circuit design and architecture. We assume α_{sc} does not depend on device and technology scale. We verify this assumption in Table III by showing the average short circuit power ratio at different Vdd and Vt levels.

benchmark	70nm Vdd=1.1 Vt=0.25		100nm Vdd=1.3 Vt=0.32		70nm Vdd=1.0 Vt=0.20	
	logic	interconnect	logic	interconnect	logic	interconnect
alu4	1.43	1.12	1.44	1.16	1.46	1.15
apex2	1.44	0.89	1.42	0.93	1.48	0.92
apex4	1.08	0.86	1.15	0.79	1.18	0.82
bigkey	0.74	1.64	0.76	1.71	0.72	1.68
clma	1.11	1.72	1.21	1.62	1.16	1.63

TABLE III

SHORT CIRCUIT POWER RATIO COMPARISON FOR DIFFERENT TECHNOLOGY SCALE, VDD AND VT. ARCHITECTURE SETTING: $N = 10, K = 4$.

For a given FPGA architecture (i.e, N and K), we profile each MCNC benchmark circuit to get the average switching activity for each resource type in the FPGA. The trace parameters α_{sc}, N_i^u and C_i^u depend only on the FPGA architecture and application benchmark set.

B. Leakage Power Model

The leakage power is modeled as follows,

$$P_{static} = \sum_i N_i^t P_i^s \tag{6}$$

For resource type i, N_i^t is the total number of circuit elements, and P_i^s is the leakage power for a type i element. Notice that usually $N_i^t > N_i^u$ because the resource utilization rate is low in FPGAs. For a FPGA architecture with power-gating capability, an unused circuit element can be power-gated to save leakage power. In that case, the total leakage power is modeled by the following formula:

$$P_{static} = \sum_i N_i^u P_i + \alpha_{gating} \cdot \sum_i (N_i^t - N_i^u) P_i \tag{7}$$

where α_{gating} is the average leakage ratio between a power-gated circuit element and a circuit element in normal operation. SPICE simulation shows that sleep transistors can reduce leakage power by a factor of 300 and $\alpha_{gating} = 0.003$ is used in this paper.

C. Delay Model

To avoid the static timing analysis for the whole circuit implemented on a given FPGA fabric, we obtain the structure of the ten longest circuit paths including the critical path for each circuit. The path structure is the number of elements of different resource types, i.e., LUT, wire segment and interconnect switch, on one circuit path. We assume that the new critical path due to different Vdd and Vt levels is among these ten longest paths found by our benchmark profiling. When Vdd and Vt change, we can calculate delay values for the ten longest paths under new Vdd and Vt levels, and choose the largest one as the new critical path delay. Therefore, the FPGA delay can be calculated as follows:

$$D = \sum_i N_i^p D_i \tag{8}$$

For resource type i, N_i^p is the number of circuit elements that the critical path goes through, and D_i is the average delay of such a circuit element. D_i is the circuit parameter depending on Vdd, Vt, process technology, and FPGA architecture. To get the path statistical information N_i^p, we only need to place and route the circuit *once* for a given FPGA architecture.

D. Validation of Ptrace

To validate Ptrace, we consider both 70nm and 100nm technology. We assume Vdd=1.0 and Vt=0.2 for 70nm technology, and Vdd=1.3 and Vt=0.32 for 100nm technology. We map 20 MCNC benchmarks to each architecture. For every architecture, power and delay are computed as the geometric mean of the 20 benchmarks. Figure 5 compares power and delay between Psim and Ptrace. Compared to cycle-accurate simulation, the average power error of Ptrace is 3.4% and average delay error is 6.1% [3]. From the figure, the Ptrace will give the same trend of power and delay as Psim. Therefore, Ptrace has a high fidelity. Moreover the run time of Ptrace is $2s$, while that of Psim is 120 hours.

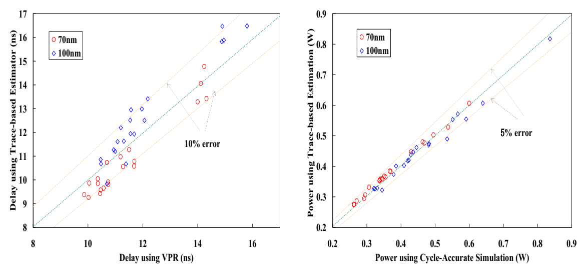

Fig. 5. Comparison between Psim and Ptrace

IV. Hyper-arch Evaluation

A. Overview

In this section, we evaluate four FPGA hyper-arch classes: $Class1$, $Class2$, $Class3$ and $Class4$. $Class1$ is the conventional FPGA using homogeneous-Vt for both interconnect and logic block (in short, homogeneous-Vt). $Class2$ applies different Vt to logic blocks and interconnects (in short heterogeneous-Vt). $Class3$ and $Class4$ are the same as $Class1$ and $Class2$, respectively, except that unused logic blocks and interconnects are power-gated. All these hyper-arch classes are summarized in Table V. We compare them with the baseline hyper-arch, which has a cluster size of 8, LUT size of 4, Vdd of 0.9v (suggested by ITRS [11]), and Vt of 0.3v that is optimized with respect with the above architecture and Vdd. The base line hyper-arch and evaluation ranges for device and architecture are shown in Table IV. Note that a high Vt is applied to all SRAM cells for configuration to reduce leakage power as suggested by [7].

In our study, we find that utilization rate of FPGA circuit (utilization rate is defined as the utilization rate of logic blocks, i.e., number of used logic blocks over total available logic blocks) does not affect the hyper-arch evaluation. As shown in Table VI, the best hyper-archs under different utilization rate are the same. Therefore throughout our following study we assume the logic block utilization rate to be 0.5.

We organize this section as follows: First, we study the low power hyper-arch in Section IV-B, then we analyze the impact of power gating in Section IV-C, and we compare the hyper-archs between different classes in Section IV-D, area and ED-tradeoff is studied in Section IV-E and the impact of device tuning and architecture tuning is studied in Section IV-F, finaly we present Vt and architecture optimization for chip-level Voltage Scaling in Section IV-G.

[3]All critical paths in experiment were among the ten longest path. The critical delay difference between Ptrace and Psim is due to that Ptrace ignores the impact of path branches that are considered in Psim

Baseline FPGA device/arch parameter values			
Vdd	Vt	N	K
0.9v	0.3v	8	4
Value range for device/arch optimization			
Vdd	Vt	N	K
0.8v-1.1v	0.2v-0.4v	6-12	3-7

TABLE IV

BASELINE HYPER-ARCH AND EVALUATION RANGES.

hyper-arch Class	Case to study
Class1	homogeneous-Vt w/o power-gating
Class2	heterogeneous-Vt w/o power-gating
Class3	homogeneous-Vt w/ power-gating
Class4	heterogeneous-Vt w/ power-gating

TABLE V

SUMMARY OF FPGA HYPER-ARCH CLASSES.

Utilization rate	0.3					0.5					0.8				
	Vdd (V)	CVt (V)	IVt (V)	(N, k)	ED (nJ· ns)	Vdd (V)	CVt (V)	IVt (V)	(N, k)	ED (nJ· ns)	Vdd (V)	CVt (V)	IVt (V)	(N, k)	ED (nJ· ns)
Class1	0.9	0.30	0.30	(6,7)	30.3	0.9	0.30	0.30	(6,7)	23.3	0.9	0.30	0.30	(6,7)	19.1
Class2	0.9	0.30	0.25	(8,5)	28.6	0.9	0.30	0.25	(8,5)	21.4	0.9	0.30	0.25	(8,5)	17.1
Class3	0.9	0.25	0.25	(12,4)	11.3	0.9	0.25	0.25	(12,4)	11.1	0.9	0.25	0.25	(12,4)	11.0
Class4	0.9	0.20	0.25	(8,4)	11.2	0.9	0.20	0.25	(8,4)	11.0	0.9	0.20	0.25	(8,4)	10.9

TABLE VI

MIN-ED HYPER-ARCH UNDER DIFFERENT UTILIZATION RATES.

B. *Low Power hyper-arch*

In this section, we present the low power hyper-arch for a given performance range to show the power reduction achieved by device and architecture co-optimization. For each benchmark, we assume it works in its highest frequency (1/critical path delay). For each hyper-arch, we compute the energy and delay as the geometric mean of 20 MCNC benchmarks. Figure 6 illustrates the energy-delay graph for hyper-archs within delay range from 9ns to 10ns. Table VII shows the maximum and minimum energy architecture within such delay range for each classes. From the table, we find that the device and architecture co-optimization can reduce energy by up to 87% compare to the maximum energy hyper-arch.

Class	Maximum energy hyper-arch						Minimum energy hyper-arch						Energy
	Vdd (V)	CVt (V)	IVt (V)	(N,K)	Energy (nJ)	Delay (ns)	Vdd (V)	CVt (V)	IVt (V)	(N,K)	Energy (nJ)	Delay (ns)	Reduction %
Class1	1.1	0.20	0.20	(12,7)	33.5	9.08	1.0	0.20	0.20	(6,5)	5.05	9.91	84.92
Class2	1.1	0.20	0.20	(12,7)	33.5	9.08	1.0	0.25	0.20	(10,6)	4.11	9.94	87.73
Class3	1.1	0.20	0.20	(12,7)	16.5	9.71	1.0	0.20	0.20	(10,6)	2.15	9.96	86.97
Class4	1.1	0.20	0.20	(12,7)	16.5	9.71	1.0	0.20	0.20	(10,6)	2.15	9.96	86.97

TABLE VII

ENERGY REDUCTION FOR HYPER-ARCHS WITHIN DELAY RANGE FROM 9NS TO 10NS.

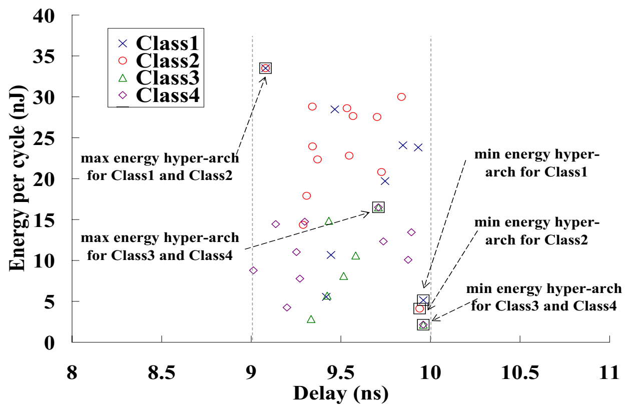

Fig. 6. Hyper-archs within delay range from 9ns to 10ns.

Table VIII shows the minimum delay and minimum energy hyper-archs for each class. We find that for the minimum delay hyper-arch, custersize is 6 and LUT size is 7, and for the minimum energy hyper-archs, LUT size is 4, cluster size is 12 for classes with power gating and is 8 or 12 for the classes without power gating. This is very similar to the previous evaluation result [4], [10].

Hyper-arch Classes	Minimum delay hyper-arch						Minimum energy hyper-arch					
	Vdd (V)	CVt (V)	IVt (V)	(N,K)	Energy (nJ)	Delay (ns)	Vdd (V)	CVt (V)	IVt (V)	(N,K)	Energy (nJ)	Delay (ns)
Class1	1.1	0.20	0.20	(6,7)	28.7	8.09	0.8	0.35	0.35	(8,4)	0.96	57.7
Class2	1.1	0.20	0.20	(6,7)	28.7	8.09	0.8	0.25	0.30	(12,4)	0.89	40.1
Class3	1.1	0.20	0.20	(6,7)	14.7	8.66	0.8	0.30	0.30	(12,4)	0.55	30.2
Class4	1.1	0.20	0.20	(6,7)	14.7	8.66	0.8	0.30	0.30	(12,4)	0.55	30.2

TABLE VIII
MINIMUM DELAY AND MINIMUM ENERGY HYPER-ARCHS.

Usually, higher performance hyper-archs will consume more power. To illustrates the tradeoff between power and delay, we introduce the concept *dominant hyper-arch*: If hyper-arch *A* has less energy consumption and a smaller delay than hyper-arch *B*, then we say that *B* is inferior to *A*. We define the *dominant hyper-arch* (in short, dom-arch) as the set of hyper-archs that are not inferior to any other hyper-archs. Figure 7 presents the energy-performance trade-off for FPGA dom-archs of *Class 1* and *Class 2*.

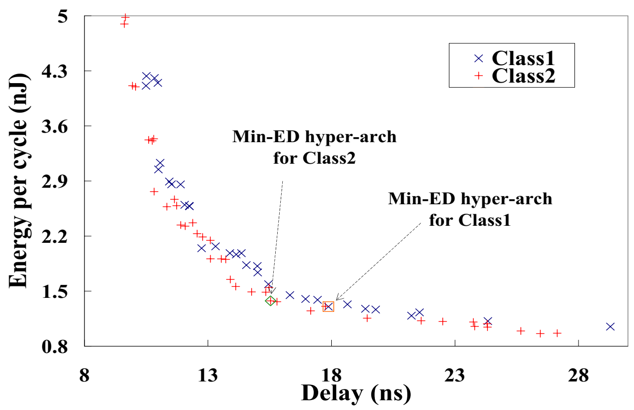

Fig. 7. Dom-archs of Class 1 and Class 2.

C. Impact of Power Gating

Power-gating can be applied to unused FPGA logic blocks and interconnect to reduce leakage power. Figure 3 presents the Vdd-gateable circuit elements. In this section, we assume a 210X PMOS sleep transistor for a logic block, a 10X PMOS sleep transistor for a 7X switch buffer, and 1X PMOS sleep transistor for connection buffer. The detailed discussion for sleep transistor sizing is in Section IV-E. The sleep transistors will introduce delay overhead for switch box and logic block. However, for connection box, because the input MUX for a logic block is no longer needed when power gating is applied, as shown in Figure 3(c), applying power gating may sometimes reduce delay. Table IX shows the normalized delay under different device setting. In the table, the delay is normalized with respect to the delay of circuits without power gating, and for the chip level delay, we assume architecture $N = 12$, $K = 4$. From the table, we observe that there are more performance lost under high performance device setting (high Vdd and low Vt) than low performance device setting (low Vdd and high Vt). Under low performance device setting, applying power gating may actually improve performance.

By applying power gating, the leakage power can be greatly reduced. SPICE simulation shows that sleep transistors can reduce leakage power by a factor of 300. Table IX shows the chip energy. We find that applying power gating can achieve significant power reduction with only small delay overhead (under low performance device setting, it can even reduce delay). Therefore, it is worth using power gating to reduce FPGA power.

Figure 8 presents the energy-performance trade-off for FPGA dom-archs of *Class 3* and *Class 4* (classes with power gating). Note that the power gap between Class3 and Class4 is smaller than that between Class1 and Class2 because leakage power is significantly reduced by field programmable power-gating and therefore the more detailed Vt tuning such as heterogeneous-Vt has a smaller impact.

Vdd (V)	Vt (V)	Switch box	Connection box	Logic block	Chip level delay	Chip level energy
0.8	0.20	108.87	92.57	102.20	99.40	24.58
0.8	0.25	104.99	80.36	102.16	97.86	35.25
0.8	0.30	104.27	62.17	101.75	96.13	49.40
0.8	0.35	103.95	42.77	101.17	91.42	59.55
0.8	0.40	103.51	26.02	101.17	90.70	63.39
0.9	0.20	109.80	102.82	103.44	101.68	25.68
0.9	0.25	105.71	94.02	102.65	100.08	38.62
0.9	0.30	104.64	77.39	102.05	99.40	54.72
0.9	0.35	104.30	57.43	101.76	93.89	66.53
0.9	0.40	103.56	38.29	101.20	90.41	71.10
1.0	0.20	110.64	113.79	104.22	103.84	28.90
1.0	0.25	106.01	103.94	103.94	102.47	41.29
1.0	0.30	104.71	91.76	102.90	102.33	57.33
1.0	0.35	104.33	72.77	102.57	97.28	72.18
1.0	0.40	103.77	53.81	102.03	93.56	78.06
1.1	0.20	111.58	122.11	105.05	105.19	40.50
1.1	0.25	109.49	113.99	104.97	104.23	48.45
1.1	0.30	105.47	103.00	104.13	104.23	57.62
1.1	0.35	104.42	89.21	103.71	100.41	67.29
1.1	0.40	104.16	70.07	103.30	96.56	75.63

TABLE IX

DELAY AND ENERGY UNDER DIFFERENT DEVICE SETTING. DELAY AND ENERGY IS NORMALIZED WITH RESPECT TO THE DELAY OF CIRCUITS WITHOUT POWER GATING. WE ASSUME $N = 12$ AND $K = 4$ FOR CHIP LEVEL DELAY AND ENERGY.

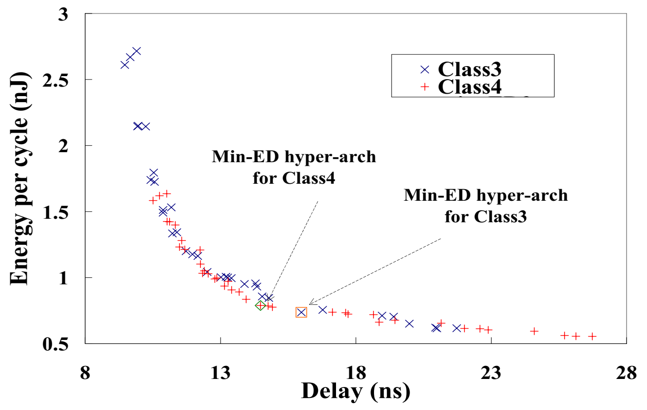

Fig. 8. Dom-archs of *Class 3* and *Class 4*.

D. Comparison Between Classes

In this section, we compare the hyper-archs between different classes. The min-ED hyper-archs for all classes are summarized in Table X. Compared to the baseline hyper-arch, *Class 2* reduces the min-ED by 20.5%. By applying heterogeneous-Vt we can reduce ED without any area increase. *Class 3* reduces the min-ED by 58.9% and *Class 4* reduces min-ED by 59.0%. Note that ED reduction for *Class 3* and *Class 4* is almost the same. This is because leakage power is greatly reduced by power-gating and therefore the more detailed Vt tuning such as heterogeneous-Vt has a smaller impact on power reduction, as discussed in Section IV-C.

Hyper-arch.Class	Vdd (V)	CVt (V)	IVt (V)	(N, k)	Energy (nJ)	Delay (ns)	ED (nJ· ns)	ED Reduction %	Normalized Area %
Baseline	0.9	0.30	0.30	(8,4)	1.19	22.6	26.9	-	100
Class1	0.9	0.30	0.30	(6,7)	1.30	17.9	23.3	13.4%	167
Class2	0.9	0.30	0.25	(8,5)	1.38	15.5	21.4	20.5%	109
Class3	0.9	0.25	0.25	(12,4)	0.74	16.0	11.1	58.9%	126
Class4	0.9	0.20	0.25	(8,4)	0.79	14.5	11.0	59.0%	144

TABLE X

COMPARISON BETWEEN BASELINE AND MIN-ED HYPER-ARCH IN *Class 1*, *Class 2*, *Class 3*, AND *Class 4*. NOTE: FOR THE HETEROGENEOUS-VT CLASSES, I.E., CLASS1 AND CLASS3, CVT=IVT.

Table XI and Table XII present the dom-archs for each class in high performance range (delay≈10ns) and low performance range (delay≈20ns), respectively. From the table, we observe the following:

1) In high performance range, the LUT size larger than that in the low performance range. This is because large LUT size will lead to smaller delay, this is similar to the previous evaluation result [4], [10].

2) There is no obvious trend for Cluster size, that means the applying Heterogeneous-Vt and power-gating has little effect on cluster size of the min-ED hyper-archs.
3) Within high performance range, the Vdd of hpyer-archs in $Class3$ and $Class4$ is higher than that of $Class1$ and $Class2$, and the Vt of hyper-archs in $Class3$ and $Class4$ is lower. However, in the low performance range, we have the inverse trend of Vdd and Vt compare to that of the high performance range. This is because in high performance range, applying power gating will increase delay, as discussed in Section IV-C, in order to achieve the same performance, higher Vdd and lower Vt are required for classes with power gating. But in the low performance range, applying power gating will reduce delay, therefore, the lower Vdd and higher Vt can be used in the classes with power gating.

Class1						Class2						
Vdd (V)	Vt (V)	(N, k)	Energy (nJ)	Delay (ns)	ED (nJ· ns)	Vdd (V)	CVt (V)	IVt (V)	(N, k)	Energy (nJ)	Delay (ns)	ED (nJ· ns)
1.0	0.20	(6,7)	5.05	9.91	50.0	1.0	0.20	0.20	(6,5)	5.05	9.91	50.3
1.0	0.20	(10,5)	5.14	9.96	51.2	1.0	0.25	0.20	(10,6)	4.11	9.94	40.9
0.9	0.20	(6,7)	4.11	10.5	43.1	1.0	0.25	0.20	(6,6)	4.10	10.1	41.3
0.9	0.20	(8,7)	4.23	10.5	44.5	0.9	0.20	0.20	(6,7)	4.11	10.5	44.5
0.9	0.20	(10,6)	4.21	10.8	45.6	0.9	0.20	0.20	(8,7)	4.23	10.5	44.5
Class3						Class4						
Vdd (V)	Vt (V)	(N, k)	Energy (nJ)	Delay (ns)	ED (nJ· ns)	Vdd (V)	CVt (V)	IVt (V)	(N, k)	Energy (nJ)	Delay (ns)	ED (nJ· ns)
1.0	0.20	(6,6)	2.14	9.93	21.3	1.0	0.20	0.25	(10,6)	2.15	9.96	21.4
1.0	0.20	(10,6)	2.15	9.96	21.4	1.0	0.20	0.25	(8,6)	2.14	10.2	21.9
1.0	0.20	(8,6)	2.14	10.2	21.9	1.0	0.20	0.25	(8,5)	1.74	10.4	18.2
1.0	0.20	(8,5)	1.74	10.4	18.9	1.0	0.20	0.25	(6,7)	1.58	10.5	17.4
1.0	0.20	(6,5)	1.79	10.5	18.9	1.0	0.20	0.25	(8,7)	1.62	10.7	17.4

TABLE XI

COMPARISON BETWEEN CLASSES IN HIGH PERFORMANCE RANGE.

Class1						Class2						
Vdd (V)	Vt (V)	(N, k)	Energy (nJ)	Delay (ns)	ED (nJ· ns)	Vdd (V)	CVt (V)	IVt (V)	(N, k)	Energy (nJ)	Delay (ns)	ED (nJ· ns)
0.9	0.30	(6,6)	1.33	18.7	24.8	0.9	0.30	0.30	(8,5)	1.28	19.4	24.7
0.9	0.30	(8,5)	1.28	19.4	24.7	1.0	0.35	0.30	(12,4)	1.16	19.5	22.5
0.9	0.30	(10,5)	1.27	19.8	25.1	0.9	0.30	0.30	(10,5)	1.27	19.8	25.1
0.9	0.30	(12,4)	1.19	21.2	25.3	0.9	0.30	0.30	(12,4)	1.19	21.2	25.3
0.9	0.30	(6,4)	1.23	21.6	26.5	0.9	0.35	0.30	(6,7)	1.12	21.6	24.3
Class3						Class4						
Vdd (V)	Vt (V)	(N, k)	Energy (nJ)	Delay (ns)	ED (nJ· ns)	Vdd (V)	CVt (V)	IVt (V)	(N, k)	Energy (nJ)	Delay (ns)	ED (nJ· ns)
0.8	0.25	(8,5)	0.71	19.0	13.5	0.9	0.25	0.30	(12,4)	0.66	18.9	12.5
0.8	0.25	(10,5)	0.70	19.4	13.7	0.9	0.25	0.30	(8,4)	0.68	19.4	13.2
0.8	0.25	(6,4)	0.65	20.0	13.0	0.8	0.25	0.25	(6,4)	0.65	20.0	13.0
0.8	0.25	(8,4)	0.62	20.9	12.9	0.8	0.25	0.25	(8,4)	0.62	20.9	12.9
0.8	0.25	(12,4)	0.62	21.0	12.9	0.8	0.25	0.25	(12,4)	0.62	21.0	12.9

TABLE XII

COMPARISON BETWEEN CLASSES IN LOW PERFORMANCE RANGE.

E. ED and Area Tradeoff

Area is important for FPGA design, especially when power-gating is applied since sleep transistors may introduce delay and area overhead. To our surprise, power-gating may reduce ED and area simultaneously because it offers a bigger solution space to explore at the chip level. Because only one sleep transistor is used for one logic block, we assume a 210X PMOS for the sleep transistor with negligible area overhead. Moreover, we observe that a 1X PMOS as the sleep transistor for one switch in connection box provides good performance, any further increase of the sleep transistor size will not improve the performance much. Therefore, we use a 1X PMOS as sleep transistor for one switch in connection box. The sleep transistors for the switches in the routing box, however, may affect delay greatly. Figure 9 and Figure 10 presents the chip-level ED-area tradeoff for different classes, considering the following sleep transistor sizes: 2X, 4X, 7X, and 10X PMOS for a 7X switch. We prune inferior solutions with both ED and area larger than any alternative solution. From the figure, we see that we can significantly reduce area with very small ED increase compare to the min-ED hyper-arch in each class. To achieve the best ED-area tradeoff, we find out the hyper-archs with minimum ED-area product, which are shown in Table XIII. From the table, we find that device and architecture co-optimization can significantly reduce both area and ED.

F. Comparison of Device and Architecture Tuning

In this section, we compare the impact of device tuning and architecture tuning. Figure 11 and Table XIV compare the impacts of device tuning and architecture tuning, where each set of data points is the hyper-archs for a given device setting. For example, set D4 is the dom-archs under Vdd=1.0 and Vt=0.25. From the figure, we observe that a change on the device

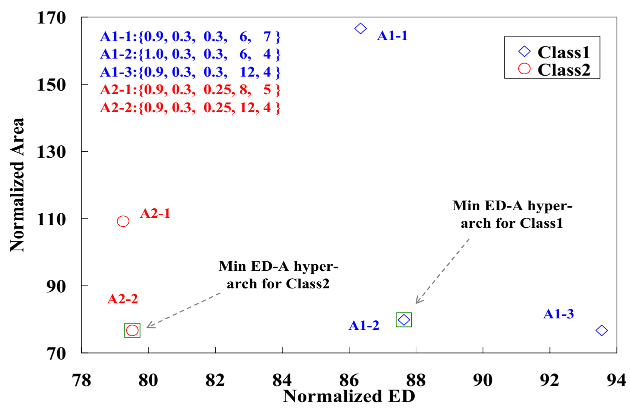

Fig. 9. ED and area trade-off for $Class1$ and $Class2$. ED and area are normalized with respect to those for the baseline architecture. G refers to the sleep transistor size for switch in the routing box.

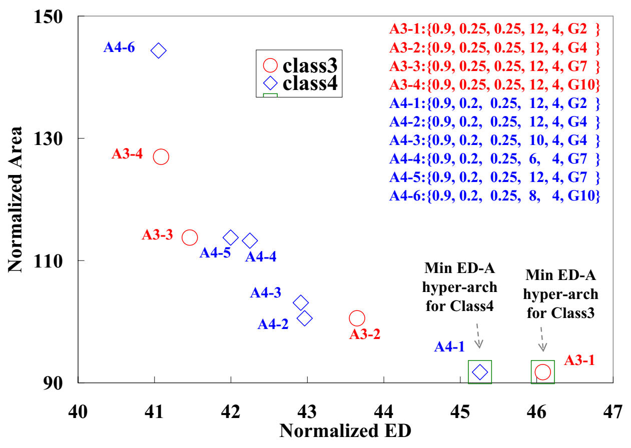

Fig. 10. ED and area tradeoff for $Class3$ and $Class4$.

	Vdd (V)	CVt (V)	Ivt (V)	(N,K)	Sleep transistor size	Normalized ED	Normalized Area	ED-Area product
Class1	1.0	0.30	0.30	(6,4)	-	87.64	79.85	69.98
Class2	0.9	0.30	0.25	(12,4)	-	79.52	76.71	70.00
Class3	0.9	0.25	0.25	(12,4)	2	46.08	91.75	42.28
Class4	0.9	0.20	0.25	(12,4)	2	45.26	91.75	41.52

TABLE XIII

MININUM ED-AREA PRODUCT HYPER-ARCHS FOR DIFFERENT CLASSES. ED, AREA, AND ED-AREA PRODUCT ARE NORMALIZED WITH RESPECT TO BASELINE.

level leads to a more significant change in power and delay than architecture change does. For example, for device setting Vdd=0.9v, Vt=0.25v, energy for different architecture is from 1.84nJ to 2.07nJ, and delay is from 12.7ns to 16.2ns. However, if we increase Vt by 0.05v, i.e., Vdd=0.9v, Vt=0.3v, the energy range is from 1.19nJ to 1.33nJ and the delay range is from 17.9ns to 21.6ns. There is no overlap of delay and energy ranges between two device settings. Therefore, it is important evaluating both device and architecture instead of evaluating architecture only.

G. Vt and Architecture Optimization for chip-level Voltage Scaling

Chip-level voltage scaling can be applied to FPGA to change Vdd level and reduce energy consumption without violating the timing specification for the current application in the just-in-time computation fashion. Given the distribution of Vdd levels for different application in an FPGA platform, the threshold voltage Vt of the chip can be optimized to minimize the weighted arithmetic mean of ED (weighted ED), where the weight is the given distribution.

Below, we assume the distribution of Vdd level 0.8v, 0.9v, 1.0v, and 1.1v to be 12.5%, 25%, 37.5%, and 25%, respectively. We find the optimal Vt for logic blocks and for interconnects to minimize the weighted mean-ED. Vt optimization is performed for the entire Vdd range from 0.8v to 1.1v, two Vdd sub-ranges $\{0.8v, 0.9v\}$ and $\{1.0v, 1.1v\}$, or each single Vdd level. Instead

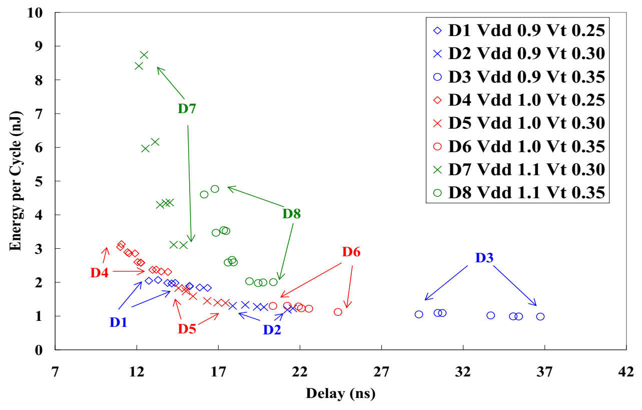

Fig. 11. Hyper-archs under different device setting.

Vdd (V)	Vt (V)	Min energy (nJ)	Max energy (nJ)	Min delay (ns)	Max delay (ns)
0.9	0.25	1.84	2.07	12.7	16.2
0.9	0.30	1.19	1.33	17.9	21.6
0.9	0.35	0.98	1.09	29.3	36.7
1.0	0.25	2.31	3.13	11.0	13.9
1.0	0.30	1.12	1.30	20.3	24.3
1.0	0.35	5.50	16.0	9.77	12.0
1.1	0.25	3.10	8.74	12.1	14.9
1.1	0.30	1.98	4.77	16.1	20.4

TABLE XIV

Power and delay ranges for different device settings.

of using uniform Vt over the entire Vdd range, using different Vt in different Vdd subranges may improve the weighted min-ED. However, using different Vt will increase design and fabrication cost because the producer may have to design different platforms for different Vdd subranges. Table XV presents the Vt and architecture optimization result. If we apply two platforms for different Vdd subranges, the weighted ED can be reduced by about 5%, and if we apply different platforms for different levels, min ED can be reduced by about 10%.

hyper-architecture Classes	Single platform (Vdd 0.8 1.1				Tow platforms							ED reduction (%)
					Vdd=0.8, 0.9			Vdd=1.0, 1.1			weighted ED	
	Cvt (V)	Ivt (V)	(N,K)	weighted ED $(nJ \cdot ns)$	Cvt (V)	Ivt (V)	(N,K)	Cvt (V)	Ivt (V)	(N,K)	$(nJ \cdot ns)$	
Class1	0.30	0.30	(6,4)	30.89	0.30	0.30	(6,7)	0.35	0.35	(12,4)	29.40	4.82
Class2	0.30	0.30	(6,4)	30.89	0.30	0.25	(8,5)	0.35	0.30	(12,5)	29.26	5.28
Class3	0.25	0.25	(12,4)	17.48	0.25	0.25	(12,4)	0.30	0.30	(6,4)	16.77	4.06
Class4	0.20	0.30	(12,4)	15.08	0.20	0.25	(8,4)	0.20	0.30	(12,4)	14.32	5.04

hyper-architecture classes	Different platforms for each Vdd level													ED reduction (%)
	Vdd=0.8			Vdd=0.9			Vdd=1.0			Vdd=1.1			Weighted ED	
	Cvt (V)	Ivt (V)	(N,K)	Cvt (V)	Ivt (V)	(N,K)	Cvt (V)	Ivt (V)	(N,K)	Cvt (V)	Ivt (V)	(N,K)	$(nJ \cdot ns)$	
Class1	0.30	0.30	(6,7)	0.30	0.30	(6,7)	0.30	0.30	(6,4)	0.35	0.35	(6,4)	28.04	9.23
Class2	0.30	0.25	(8,5)	0.30	0.25	(8,5)	0.35	0.30	(12,4)	0.35	0.35	(6,4)	26.58	13.96
Class3	0.20	0.20	(6,4)	0.25	0.25	(6,4)	0.25	0.25	(6,4)	0.35	0.35	(12,4)	15.68	10.30
Class4	0.20	0.25	(8,4)	0.20	0.25	(8,4)	0.20	0.25	(12,4)	0.20	0.35	(12,4)	13.74	8.88

TABLE XV

Vt and architecture optimization.

V. Conclusions and Discussions

In this paper, we have developed trace-based power and performance models for FPGA. The new models are much faster but yet accurate compared to the cycle-accurate simulation [4]. The one-time use of cycle-accurate simulation is applied to collect the timing and power trace for given benchmark set and given FPGA architecture. Then the trace can be re-used to calculate timing and power via closed-form formulae for different device parameters and technology scaling.

Using the trace-based estimation, we have first performed device (Vdd and Vt) and architecture (cluster and LUT size) co-optimizations for low power FPGAs. We assume the 70nm ITRS technology and use the following baseline for comparison: Vdd of 0.9v as suggested by ITRS, Vt of 0.3v as given by our Vt optimization for min-ED (i.e., minimum energy delay product), cluster size of 8 and LUT size of 4 as in Xilinx FPGA. Compared to the baseline case, simultaneous optimization of FPGA architecture, Vdd and Vt reduces the min-ED by 12.3% and area by 20.1% for FPGA using homogeneous-Vt for the logic and interconnect without power gating, and optimizing Vt separately (i.e., heterogeneous-Vt) for the logic and interconnect reduces min-ED by 20.5% and area by 20.5%. Furthermore, power gating unused logic and interconnect reduces the min-ED by up to 57.5% and reduce area by 8.3%. Compared to the homogeneous-Vt FPGAs, the min-ED hype-arch using heterogeneous-Vt has a smaller LUT size. In addition, device (i.e., Vdd and Vt) tuning has a more significant impact on power and delay than architecture tuning does.

Assuming FPGA chip-level Vdd scaling for just-in-time computation, we have then compared (i) one fixed and optimal Vt and architecture for the entire Vdd scaling range and (ii) two optimal Vt and architectures for the two Vdd scaling subranges, (iii) different optimal Vt and architectures for different Vdd levels. Experiments show that finding optimal Vt and architectures for two subranges reduces the weighted min-ED by about 5%, and optimal Vt and architectures for different Vdd levels reduces the weighted min-ED by about 10%, where the weight is the distribution of Vdd levels used for applications with different timing requirements.

REFERENCES

[1] J. Rose, R. J. Francis, D. Lewis, and P. Chow, "Architecture of field-programmable gate arrays: The effect of logic functionality on area efficiency," *Proc. IEEE Int. Solid-State Circuits Conf.*, 1990.

[2] J. Kouloheris and A. E. Gamal, "FPGA area vs. cell granularity - lookup tables and PLA cells," in *1st ACM Workshop on FPGAs, berkeley, CA*, Feb 1992.

[3] 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.

[4] 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.

[5] 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.

[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] 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.

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

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

[10] Y. Lin, F. Li and L. He, "Circuits and architectures for vdd programmable FPGAs," in *Proc. ACM Intl. Symp. Field-Programmable Gate Arrays*, Feb 2005.

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

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

[13] 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.