Device And Architecture Co-Optimization for FPGA Power Reduction *

Lerong Cheng, Phoebe Wong, Fei Li, Yan Lin, and Lei He

Electrical Engineering Department
University of California, Los Angeles, CA
{lerong, phoebe, feil, ylin, lhe}@ee.ucla.edu, http://eda.ee.ucla.edu

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% without any chip area increase 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.

Categories and Subject Descriptors: B.7.1 [Integrated Circuits]: Types and Design Styles **General Terms:** Performance, Design **Keywords:** FPGA, low power, power-gating, Ptrace, Psim

1. 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, 2, 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, 2, 3, 4, 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).

*This paper is partially supported by NSF CAREER award CCR-0093273/0401682 and NSF grant CCR-0306682. Address comments to lhe@ee.ucla.edu.

DAC 2005, June 13–17, 2005, Anaheim, California, USA.

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% without additional area. 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 2 presents the trace-based estimation models. Section 3 performs the architecture and device co-optimization. Section 4 concludes this paper. More details and Vt optimization for chip level voltage scaling are included in the technical report [12].

2. TRACE-BASED ESTIMATION

In this section, we first discuss the preliminaries of FPGA architecture and review the power model used in the cycle-accurate simulation [4]. We then present and validate our trace-based estimation called Ptrace for FPGA power and delay.

2.1 Preliminaries

We assume the same cluster-based island style FPGA as previous work [3, 4]. A logic block is a cluster of fully connected Basic Logic Elements (BLEs), and the cluster size N is the number of BLEs in a logic block. Each BLE consists of one Lookup Table (LUT) and one flip-flop. For an island style routing structure, logic blocks are surrounded by programmable routing channels, and the routing wires in both horizontal and vertical channels are segmented by *routing switch blocks*. In this paper, we use a fixed routing architecture, i.e., fully buffered routing switches and uniform wire segment spanning 4 logic blocks; and we study the impact of the N and K on architecture optimization. Moreover, we assumed the routing channel width (number of tracks in each routing channel) to be 1.2 times of the minimum routing channel width (the minimum width to let the FPGA circuit be routeable). Because there is a limited number of cluster size and LUT size combinations, the previous evaluation method based on cycle-accurate simulation can be applied when only architecture optimization is considered.

We define our baseline FPGA as the cluster-based island style FPGA architecture with a Vdd of 0.9v suggested by ITRS [11] at 70nm technology, LUT size of 4 and cluster size of 8 as the Xilinx FPGA, and a Vt of 0.3v, which is optimized by our Vt tuning for minimum ED product. If we use a Vt of 0.35V, the ED increases by 58%. This illustrates the benefit of Vt optimization and the quality of the baseline FPGA. Table 1 gives Vdd and Vt levels for the baseline FPGA and the evaluation ranges of Vdd, Vt, N and K.

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 1: Baseline hyper-arch and evaluation ranges.

2.2 Cycle-Accurate Simulation

Given the above FPGA architecture, a detailed power model has been proposed for cycle-accurate simulation (in short *Psim*) [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 during a signal transition. The switching power is due to the charging and discharging of load capacitance, and can be modeled as follows,

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

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.

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

where t_r is the signal transition time and $\alpha_{sc}(t_r)$ is the ratio between short-circuit power and switching power, and it 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 running 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 used to get the switching activity as well as signal transition time.

2.3 Trace-based Estimation

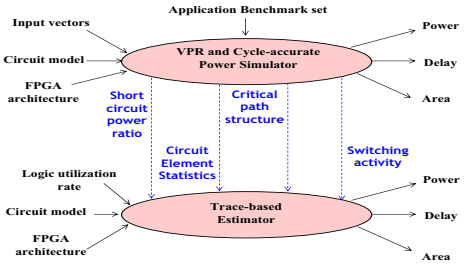

Figure 1: Cycle-accurate simulation versus trace-based estimation.

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 2: Trace information, device and circuit parameters.

The cycle-accurate simulation is time consuming because a large number of the input vectors needs to be simulated using a 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 hyper-arch combinations can be easily over a few hundreds. We develop a runtime efficient trace-based estimation method (in short *Ptrace*). 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 1 illustrates the relation between the cycle-accurate simulation and trace-based estimation. Table 2 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; circuit parameters, including P_i^s, C_i^u, and D_i, are what depend on circuit design and device. The details of Ptrace is discussed in the following.

2.3.1 Dynamic Power Model

Dynamic power includes switching power and short-circuit power. A circuit implemented on an 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:

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

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,

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

For the type i circuit elements, C_i^u is the average load capacitance of a used circuit elements, which is averaged 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 3 by showing the average switching activity of five benchmarks at different Vdd and Vt levels.

bench-mark	70nm Vdd=1.1 Vt=0.25		100nm Vdd=1.3 Vt=0.32		70nm Vdd=1.0 Vt=0.20	
	logic	inter-connect	logic	inter-connect	logic	inter-connect
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 3: Switching activity comparison for different technology scale, Vdd and Vt.

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

$$P_{sc} = P_{sw} \cdot \alpha_{sc} \quad (5)$$

Where α_{sc} is the ratio between short circuit power and switching power. Such ratio is a circuit parameter depending on FPGA circuit design and architecture. We assume α_{sc} does not depend on device and technology scale.

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 the benchmark set.

2.3.2 Leakage Power Model

The leakage power is modeled as follows,

$$P_{static} = \sum_i N_i^t P_i^s \quad (6)$$

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 an 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:

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

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 $\alpha_{gating} = 1/300$ is used in this paper.

2.3.3 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:

$$D = \sum_i N_i^p D_i \quad (8)$$

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.

2.3.4 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 2 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% [1]. 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.

[1] 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

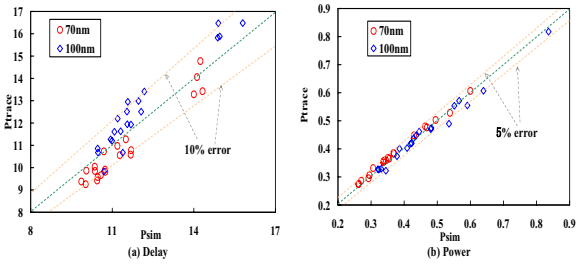

Figure 2: Comparison between Psim and Ptrace

3. HYPER-ARCH EVALUATION

3.1 Overview

In this section, we evaluate four FPGA hyper-arch classes: *Class*1, *Class*2, *Class*3, and *Class*4 (see Table 4). *Class*1 is the conventional FPGA using homogeneous-Vt for both interconnect and logic block (in short, homogeneous-Vt). *Class*2 applies different Vt to logic blocks and interconnects (in short heterogeneous-Vt). *Class*3 and *Class*4 are the same as *Class*1 and *Class*2, respectively, except that unused logic blocks and interconnects are power-gated. We compare them with the baseline hyper-arch, which together with the evaluation ranges for device and architecture are shown in Table 1. 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 chip (defined as number of used logic blocks over the total available logic blocks) does not affect the hyper-arch evaluation. Therefore throughout our following study we assume the logic block utilization rate to be 0.5. We study the effect of heterogeneous-Vt and power-gating in Section 3.2, then compare the impact of device tuning and architecture tuning in Section 3.3 .

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 4: Summary of FPGA hyper-arch Classes.

3.2 Heterogeneous-Vt and Power-gating

In this section, we present the hyper-arch evaluation. For each hyper-arch, we compute the energy and delay as the geometric mean of 20 MCNC benchmarks. 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 3 presents the energy-performance tradeoff for FPGA dom-archs. The min-ED hyper-archs for all classes are summarized in Table 5. The optimal Vt for logic blocks (CVt) is lower than the Vt for interconnects (IVt) because the interconnect leakage is more significant than logic block leakage. Compared to the baseline hyper-arch, *Class 1* reduces the min-ED by 13.7% and *Class 2* reduces the min-ED by 20.5%. Applying heterogeneous-Vt reduce ED without area increase.

Class	Vdd (V)	CVt (V)	IVt (V)	(N, k)	ED (nJ· ns)	Area
Baseline	0.9	0.30	0.30	(8,4)	26.9 (0%)	1
Class1	0.9	0.30	0.30	(6,7)	23.3 (13.4%)	1.67
Class2	0.9	0.20	0.25	(8,4)	21.4 (20.5%)	1
Class3	0.9	0.25	0.25	(12,4)	11.1 (58.9%)	1.26
Class4	0.9	0.20	0.25	(8,4)	11.0 (59.0%)	1.44

Table 5: Comparison between baseline and min-ED hyper-arch in *Class 1*, *Class 2*, *Class 3*, and *Class 4*. Note: for the homogeneous-Vt classes, i.e., Class1 and Class3, CVt=IVt.

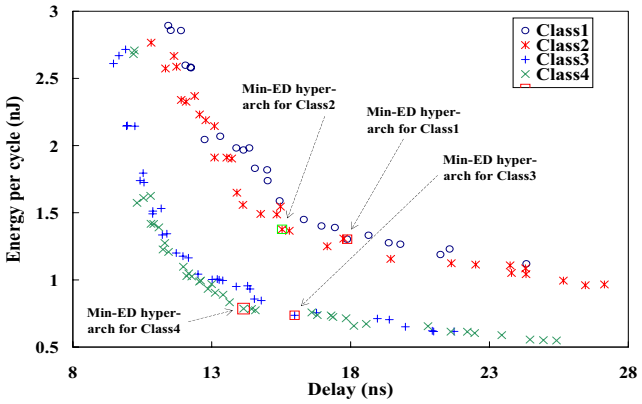

Figure 3: Delay-energy trade-off of Dom-archs.

Furthermore, power-gating can be applied to unused FPGA logic blocks and interconnect to reduce leakage power. Compared to the baseline hyper-arch, *Class 3* reduces the min-ED by 58.9% and *Class 4* reduces min-ED by 59.0%, as shown in Table 5. From Figure 3, we find 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.

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 transistor for one switch in the routing box, however, may affect delay greatly. Figure 4 presents the chip-level ED-area tradeoff for *Class*3 and *Class*4, 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. Compared to the baseline architecture, hyper-arch {Vdd=0.9, CVt=IVt=0.25, N=12, K=4} in *Class*3 reduces ED by 53.9%, and hyper-arch {Vdd=0.9, CVt=0.2, IVt=0.25, N=12, K=4} in *Class*4 reduces ED by 54.7%. Both have 2X sleep transistor for one switch in routing box but reduce area by 8.3% because the optimized cluster size is now bigger than the one in the baseline architecture. A higher ED reduction can be reached for a slightly more area.

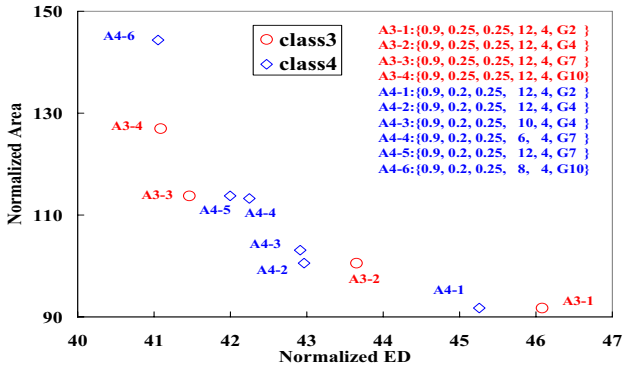

Figure 4: ED and area tradeoff for *Class*3 and *Class*4. ED and area are normalized with respect to those for the baseline architecture. G refers to the sleep transistor size for switch in the routing box.

Table 6 summarizes a few hyper-archs within a similar range of delay (about 20ns) for the 4 classes. From the table, we observe that heterogeneous-Vt decreases the LUT size in the min-ED hyper-arch. For the min-energy hyper-arch within the delay range, CVt (Vt for logic) is lower than IVt(Vt for interconnect). This causes the logic power to increase. Therefore, to compensate for the power increase, a smaller LUT size is used to reduce the logic power.

3.3 Comparison of Device and Architecture Tuning

Figure 5 and Table 7 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 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.

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 7: Power and delay ranges for different device settings.

Class1						Class2						
Vdd (V)	Vt (V)	(N, K)	Energy (nJ)	Delay (ns)	ED (nJ· ns)	Vdd (V)	CVt (V)	IVt	(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	0.35	(12,4)	1.23	18.9	23.2
0.9	0.30	(8,5)	1.28	19.4	24.7	0.9	0.30	0.35	(10,4)	1.19	18.9	22.5
0.9	0.30	(10,5)	1.27	19.8	25.1	0.9	0.30	0.35	(6,4)	1.16	20.1	23.3
0.9	0.30	(12,4)	1.19	21.2	26.5	0.9	0.30	0.35	(12,4)	1.14	20.5	23.7
0.9	0.30	(6,4)	1.23	21.6	26.5	0.9	0.30	0.35	(8,4)	1.09	22.1	24.1
Class3						Class4						
Vdd (V)	Vt (V)	(N, K)	Energy (nJ)	Delay (ns)	ED (nJ· ns)	Vdd (V)	CVt (V)	IVt	(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 6: Comparison between Classes in similar performance range.

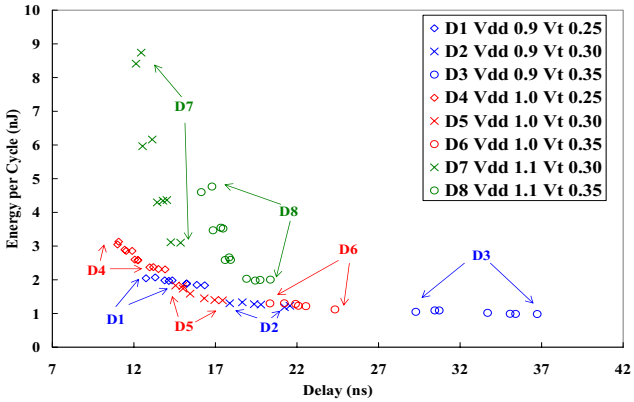

Figure 5: Hyper-archs under different device setting.

4. CONCLUSIONS AND DISCUSSIONS

In this paper, we have developed trace-based power and performance models, called Ptrace, for FPGA. Ptrace is much more efficient but yet accurate compared to the cycle-accurate simulation, Psim [4]. The one-time use of Psim is applied to collect the timing and power trace for a given benchmark set and given FPGA architectures. 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 Ptrace, 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 13.4% 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%. Furthermore, power-gating unused logic and interconnect reduces the min-ED by 54.7% and area by 8.3%. Compared to the homogeneous-Vt FPGAs, the min-ED hyper-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.

5. 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] L. Cheng, P. Wong, F. Li, Y. Lin, and L. He, "Device and architecture co-optimization for FPGA power reduction," Tech. Rep. 05-258, UCLA Engr.