Enabling Energy Efficiency in Via-Patterned Gate Array Devices

R. Reed Taylor
Carnegie Mellon University
Department of ECE
Pittsburgh, PA 15217
rt2i@ece.cmu.edu

Herman Schmit
Tabula, Inc.
444 Castro St., Suite 1120
Mountain View, CA 94303
herman@tabula.com

ABSTRACT

In an attempt to enable the cost-effective production of low- and mid-volume application-specific chips, researchers have proposed a number of so-called structured ASIC architectures. These architectures represent a departure from traditional standard-cell-based ASIC designs in favor of techniques which present more physical and structural regularity. If structured ASICs are to become a viable alternative to standard cells, they must deliver performance and energy efficiency which is competitive with standard-cell-based design techniques. This paper focuses on one family of structured ASICs known as via-patterned gate arrays, or VPGAs. In this paper, we present circuit structures and power optimization algorithms which can be applied to VPGA chips in an effort to reduce their operational power dissipation.

Categories and Subject Descriptors

B.6.3 [**Design Aids**]: Optimization; B.7.1 [**Types and Design Styles**]: Gate Arrays

General Terms

Design, Performance, Algorithms

Keywords

Structured ASIC, VPGA, Low-Power, Voltage Scaling, Power Optimization

1. INTRODUCTION

Integrated circuit manufacturing processes with feature sizes below 100nm pose a number of technological challenges to the ASIC designer. In particular, these new processes present increased manufacturing costs, substantial process variation, and exceedingly complex design rules compared to their technological predecessors.

In an attempt to mitigate these factors and to enable the cost-effective production of low- and mid-volume application-specific chips, researchers have proposed a number of so-called structured ASIC architectures. These architectures represent a departure from traditional standard-cell-based ASIC designs in favor of techniques which present more physical and structural regularity. This regularity can be leveraged in an effort to compensate for process variations, complex design rules, and rising manufacturing costs. This same regularity, however, strips structured ASICs of the power-performance flexibility which can be exploited (as in standard-cell ASIC design flows) in an effort to conserve power.

If structured ASICs are to become viable low-cost replacements for standard cell ASICs, they must deliver performance and energy efficiency which can compete with contemporary standard-cell-based designs. In [3], we introduce a number of circuits which can restore this flexibility to via-patterned gate arrays (VPGAs) using gate sizing and voltage scaling techniques. Section 2 presents one of these flexible circuit structures and briefly describes the VPGA architecture.

Section 3 of this paper presents POGA, our algorithm for exploiting this newfound power-performance flexibility in an effort to reduce the operational power consumption of structured ASIC circuits. Finally, section 4 presents results of the application of this technique to hardware benchmarks.

2. CIRCUIT STRUCTURES

While the range of structured ASIC architectures is still under active exploration, this paper will use the via-patterned gate array, or VPGA, as a prototypical structured ASIC architecture. The VPGA model considered here is heterogeneous, consisting of 3-input lookup tables (3-LUTs) as well as some high-performance logic gates (e.g. NAND). An array of these logic cells makes up the computational fabric of our VPGA. The findings relevant to this class of VPGAs should be extensible to many other structured ASIC architectures, regardless of the arrangement and internal structure of their logic cells.

2.1 A Structured ASIC Family: the VPGA

The VPGA architecture, which is formally introduced in [1] and [4], resembles that of a traditional FPGA in a number of ways: they are both generally conceived as two-dimensional arrays of programmable logic units which can

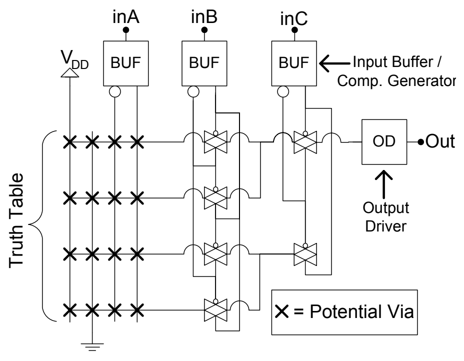

Figure 1: VPGA 3-LUT Logic Block

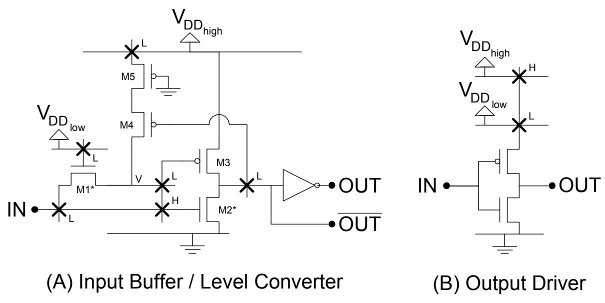

Figure 2: Dual-Supply Level Converter and Output Driver

be selectively connected through the use of a fixed routing architecture with programmable switchboxes at possible junction points.

Unlike FPGAs, VPGAs are not field-programmable devices. VPGAs are "programmed," or patterned, during latter stages of the manufacturing process by the selective placement of inter-layer vias. These vias form the required logical connections to complete and configure the device.

An example of via-patterning can be seen in Figure 1, where the truth-table for the logical behavior of a LUT is determined entirely though the placement of vias on the potential locations, marked with "X". The complement-generating input buffers are marked "BUF", and the output driver is marked "OD".

Programming through selective via placement makes VPGAs much more dense, efficient, and high-performance than FPGAs, which are programmed using SRAM cells and passgates. In fact, VPGAs exhibit performance which is comparable to that of contemporary standard-cell ASICs (SC-ASICs).

2.2 Circuit Structures for Enabling Power Optimization

Modern standard-cell libraries frequently contain several instances of each logic function with differing gate sizes, allowing the drive strength of each individual gate to be matched to its particular capacitive load. This *power-performance flexibility* is leveraged by power optimization tools to reduce operational power consumption without sacrificing design performance.

By contrast, the fixed underlying structure of an unmodified VPGA does not present any such flexibility. In order to enable power optimization in VPGA design flows, circuits which convey power-performance flexibility must be explicitly added to the VPGA logic block. In [3], several such structures are proposed.

In this paper, we will perform power optimization using *selective voltage scaling*, with dual independent voltage supplies. In selective voltage scaling, gates which drive networks that are not critical to the overall performance of a design are modified to operate with a reduced voltage supply. These modified gates will exhibit reduced performance while consuming substantially less dynamic and static power.

In order to enable power optimization through selective voltage scaling in a VPGA, the needed ingredients are: a circuit which can buffer and possibly perform level conversion on the inputs (this circuit is labeled "BUF" in Figure 1); and an output driver which can produce either high- or low-swing signals (labeled "OD" in Figure 1).

2.2.1 Input Buffer

The application of selective voltage scaling will typically create many points in a circuit where signals with differing voltage swings will interact. At these points, there is a risk of static current flow, which would rapidly undermine any achieved power savings. This static current must be prevented with the use of voltage level converters.

In the past, level converters were typically constructed from DCVS logic; however, in [2], some newer and more sophisticated circuit structures for voltage level conversion were introduced. These circuits were shown to be faster and more efficient than their DCVSL counterparts. One such circuit, modified and optimized for use in a VPGA environment, can be seen in Figure 2(A). This circuit implements the "BUF" function, as seen in Figure 1.

In this circuit, when vias are placed so as to make connections at the locations marked "L", the transistors M1, M4, and M5 all work to perform level conversion on the input. (Transistors M1 and M2, marked with "*", should be low-threshold devices.) The performance of the input buffer in this mode is acceptable; however, in a mixed-swing environment, it will often be the case that an input to a logic block will, in fact, be a full-swing signal. In this case, we would like to use via patterning to eliminate the performance penalty associated with level conversion.

This can be accomplished by instead placing a via at the location marked "H". In this case, the transistors M1, M4, and M5 will all be disconnected from our circuit and from the voltage supply rails; therefore, they will have no impact on performance. Configured in this manner, the input buffer looks like a simple series of two inverters, both powered by $V_{DD_{high}}$.

2.2.2 Output Driver

The dual-swing output driver corresponding to the "OD" block in Figure 1 is shown in Figure 2(B). Configuring this circuit is very simple: by placing a via at the locations marked either "H" or "L", it can be configured to use either $V_{DD_{high}}$ or $V_{DD_{low}}$ as its supply, producing a corresponding high- or low-swing output.

At the circuit level, both of these structures are quite compact; however, it should be noted that there will be additional overhead in terms of area due to the need to distribute $V_{DD_{low}}$, the reduced power supply. Nonetheless, these circuits will permit the VPGA logic blocks to be independently and individually configured for various power-performance modes of operation solely through the selective placement of vias.

3. POGA: POWER OPTIMIZATION FOR GATE ARRAYS

The remaining element needed to reduce the power consumption of a VPGA is an algorithm which can perform power optimization by selecting gates for modification.

In [5], an algorithm for standard-cell ASIC power optimization through selective voltage scaling, known as *clustered voltage scaling*, or CVS, was introduced. CVS was reported to save an average of 47% of the power dissipated by various applications.[6] However, CVS made a priority of clustering scaled components together so as to better match the row-based organization of SC-ASICs, and to reduce the number of voltage level conversions.

Unlike SC-ASICs, structured ASICs like VPGAs greatly loosen these constraints on placement thanks to their fixed, regular structure. By using the logic cell proposed in Section 2, all the circuitry necessary to perform level conversion will be present in every block; thus, there will be no incremental impact on area as a result of performing level conversion in any particular location. Our approach should therefore be permitted to select nets for scaling based solely on their relative contributions to overall power dissipation and path delay.

The approach we propose is known as *power optimization for gate arrays*, or POGA. Unlike CVS, POGA does not attempt to cluster scaled components together. This enables the consideration of many more complex arrangements of low- and high-swing cells than was possible with CVS.

This lifted constraint also simplifies the algorithmic task which POGA must perform and enables POGA to consider power optimization schemes which are not based around voltage scaling at all. (For example, structures which emulate gate scaling rather than voltage scaling are equally applicable to POGA, and are described in [3].) However, for clarity, in this paper we consider only the type of power optimization enabled by the circuits described in Section 2.

3.1 Underlying Formulations

Let us refer to each distinct type of hardware structure which exists on a VPGA as a *component*. The VPGA logic block described in Section 2 is an example of a component, as would be an interconnect buffer, a switchbox, an I/O block, etc. A VPGA can thus be thought of as a 2-dimensional array of such components, arranged according to some regular pattern. In addition, let us refer to the logical net which is driven by some particular on-chip component as a *node*. We will refer to the component driving a node x as $\mathcal{D}[x]$.

Then, let S be the set of all "scalable" components in a VPGA architecture; that is, the set of all components which are capable of outputting both high- and low-swing signals, according to their via-patterned configuration (to be selected by POGA). The basic logic blocks and interconnect buffers might be in S, whereas I/O blocks and other non-scalable components would not be in S.

From this, let us define a "scalability" function $\mathcal{S}[x]$ such that:

$$\mathcal{S}[x] = \begin{cases} 1 & \text{if } \mathcal{D}[x] \in S \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

In the preceding sections, this paper has effectively focused on developing the set of scalable components S by proposing circuit structures which can both accept and produce low- and high-swing signals. In general, the more on-chip components that are scalable, the more nets in a design will be eligible for scaling by POGA.

Finally, let us define:

$$\begin{aligned} \mathcal{T}[x] &= \text{the negative timing slack on node } x & (2) \\ C &= \text{the set of all critical nodes} & (3) \end{aligned}$$

where *critical nodes* are defined as the nodes comprising the transitive fan-in of the latest-arriving signal in the design.

Let N be the set of all nodes in a particular design. $N_U \subset N$ will be the set of all unscaled (high-swing) nodes, and $N_S \subset N$ will be the set of all scaled (low-swing) nodes.

3.2 POGA Specification

The fundamental task of POGA, then, is to find an assignment of nodes to N_S and N_U which reduces the power consumption of the design. First, a traditional tool flow is used to place and route the circuit. During the techmapping, placement, and routing passes, only full-swing components are used. This produces a baseline design implementation which is assumed to offer the highest available performance, with the maximum power consumption.

The POGA algorithm is a greedy approach to power savings. POGA operates by migrating an entire design to low-swing operation, and then restoring nodes to full-swing successively until acceptable performance is restored. This approach is guaranteed to terminate successfully, as, in the worst case, it will finish with a design which is identical to the baseline implementation (and which saves no power). The challenge for POGA, then, is to restore this performance while migrating the smallest possible set of components to full-swing operation.

POGA initially considers all nodes as being scaled, so $(N_S = N)$ and N_U is empty. $\mathcal{D}[x]$ is set to low-swing operation for all nodes.

At each iteration, the set of critical nodes C is identified. The best candidate node for restoration, n, is selected, such that:

$$\begin{aligned} n &\in C \\ n &\in N_S \text{ and } \mathcal{S}[n] = 1 \\ \mathcal{T}[n] &\text{ is maximal for all nodes in } C \end{aligned}$$

In other words, n is the scaled node in C with the largest negative slack. This node n is migrated from N_S to N_U, and $\mathcal{D}[n]$ is restored to high-swing operation. The timing for the design is recalculated, and this process is iterated until the entire design meets the established baseline performance, at which point POGA is finished.

3.3 Optimizations

Several optimizations not central to the operation of POGA have been implemented. We will discuss those modifications to the basic algorithm in this section.

First, the recalculation of timing and selection of the best node n can be a computationally intensive operation. As a result, the runtime of POGA can suffer in designs with large numbers of nets due to the frequent recalculations. To address this, POGA can be modified to perform the recalculation of timing only once in every Q iterations. In effect, this will cause Q nodes to be migrated from the set of critical nodes C during each iteration, before the contents of C and the slack values $\mathcal{T}[x]$ are updated.

In large designs, as Q is increased, the runtime of POGA can be drastically improved. However, when $(Q > 1)$, there will be a tradeoff in the quality of the result obtained by POGA: nodes which are no longer truly in C, or which no longer have the maximal value of $\mathcal{T}[x]$, may be needlessly restored. This can cause POGA to achieve decreasingly optimal assignments of nodes to N_S and N_U.

In the worst case, POGA will needlessly restore as many as $(Q - 1)$ nodes to full-swing operation during each iteration. Fortunately, because the power impact of restoring any single net in a design is generally very small, the overall effect of this will likely be acceptable as long as $(Q \ll \mathit{count}[N])$ (i.e. several orders of magnitude less than the number of nodes).

3.4 Implementation

In order to test the effectiveness of this algorithm as well as the dual-swing VPGA logic blocks, we implemented POGA within the Dolphin Physical Design System from Monterey Design, Inc.

In addition, exhaustive characterization of the VPGA cells (implemented in a commercial 0.13μ technology) was performed using a tool from Magma Design Automation known as SiliconSmart CR. Cells were individually characterized over a range of input slopes and load capacitances. Timing characterizations were performed for both low- and high-swing inputs, and detailed switching, short-circuit, and leakage power information was gathered. The VPGA logic block was characterized in an XOR configuration, along with high-speed NAND gates, inverters, and a buffer. Each cell ultimately had four separate models: one for each combination of low- and high-swing inputs and outputs.

Within Dolphin, the benchmark designs were first placed and routed using only high-swing nodes. Once this was completed, the baseline performance measurements were taken, following which all nodes were migrated to low-swing operation. POGA was then allowed to proceed, terminating as soon as the baseline performance had been restored.

Within the Dolphin POGA flow, when a particular node is switched to low-swing operation, its library model is simply replaced with the model of its low-swing counterpart. In addition, the models of all nodes driven by the node in question are replaced to reflect the changing needs for level conversion, and to incorporate the resulting power and performance impact. When a node is restored to high-swing mode, the models of all driven nodes are similarly updated.

In general, POGA proceeds through the netlist with no specific regard to the circuit topology. Nodes are selected for restoration based solely on their slack values, and their presence in the set C of critical nodes. The adaptable nature of the dual-swing circuits presented in Section 2 permits POGA to take this simple approach (without specifically clustering scaled nodes together), and still achieve good results.

4. RESULTS

POGA was run for a set of hardware benchmarks, and power consumption estimates were made using Dolphin's internal power estimation system (which uses circuit topology, input switching frequencies, and power characteristics gathered from SiliconSmart CR to accurately estimate switching, leakage, and short-circuit power consumption).

Benchmark	*Gates*	*Q*	*Power Savings*		
			leak	*short*	*overall*
alu	715	5	89.4%	1.3%	33.2%
firewire	2574	25	56.8%	46.9%	21.2%
mfpa	20981	500	61.7%	65.3%	35.4%
bnode	64521	250	45.2%	53.1%	25.4%

Table 1: POGA Benchmark Results

The results of running POGA on a set of hardware benchmarks are presented in Table 6. An average of 28.8% of the overall power dissipation was saved for each benchmark. Savings results for leakage and short-circuit power are presented individually; there was little overall impact on the switching power consumed (less than ±2% in each case).

5. CONCLUSIONS

The circuit structures presented in this paper, used in conjunction with the POGA algorithm, can save substantial amounts of operational power without sacrificing performance. VPGA and other structured ASIC architects should strongly consider the inclusion of circuits which convey power-performance flexibility so that power recovery techniques like POGA can be applied.

6. ACKNOWLEDGMENTS

This work was funded by the Pittsburgh Digital Greenhouse.

7. REFERENCES

[1] L. Pileggi, A. Strojwas, P. Gopalakrishnand, V. Kheterpal, A. Koorapaty, C. Patel, V. Rovner, and K. Tong. Exploring regular fabrics to optimize the performance-cost trade-off. In *Proceedings of the 40th ACM/IEEE Design Automation Conference*, pages 782–787. ACM Press, 2003.

[2] R. Puri, L. Stok, J. Cohn, D. Kung, D. Pan, D. Sylvester, A. Srivastava, and S. Kulkarni. Pushing ASIC performance in a power envelope. In *Proc. IEEE/ACM Design Automation Conference*, pages 788–793, June 2003.

[3] R. R. Taylor and H. Schmit. Creating a power-aware structured ASIC. Technical Report CSSI 04-02, The Center for Silicon Systems Implementation (CSSI), Carnegie Mellon University, March 2004.

[4] K. Tong, V. Kheterpal, V. Rovner, and L. Pileggi. Regular logic fabrics for a via patterned gate array (vpga). In *Custom Integrated Circuits Conference, Proceedings of the IEEE*, September 2003.

[5] K. Usami and M. Horowitz. Clustered voltage scaling for low-power design. In *Proceedings of the International Symposium on Low Power Design (ISLPD 95),*, pages 3–8, April 1995.

[6] K. Usami, M. Igarashi, F. Minami, T. Ishikawa, K. M, M. Ichida, and K. Nogami. Automated low-power technique exploiting multiple supply voltages applied to a media processor. *IEEE Journal of Solid-State Circuits*, 33(3):463–472, March 1998.