Journal of
Low Power Electronics
Vol. 2, 113–120, 2006

Power – Performance Optimization for Custom Digital Circuits

Radu Zlatanovici and Borivoje Nikolić*

Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720, USA

(Received: 23 December 2005; Accepted: 23 January 2006)

This paper presents a modular optimization framework for custom digital circuits in the power – performance space. The method uses a static timer and a nonlinear optimizer to maximize the performance of digital circuits within a limited power budget by tuning various variables such as gate sizes, supply, and threshold voltages. It can employ different models to characterize the components. Analytical models usually lead to convex optimization problems where the optimality of the results is guaranteed. Tabulated models or an arbitrary timing signoff tool can be used if better accuracy is desired and although the optimality of the results cannot be guaranteed, it can be verified against a near-optimality boundary. The optimization examples are presented on 64-bit carry-lookahead adders. By achieving the power optimality of the underlying circuit fabric, this framework can be used by logic designers and system architects to make optimal decisions at the microarchitecture level.

Keywords: Power – Performance Optimization, Convex Optimization, CMOS, Static Timing, Timing Models.

1. INTRODUCTION

Integrated circuit design has seamlessly entered the power-limited scaling regime, where the traditional goal of achieving the highest performance has been displaced by optimization for both performance and power. Achieving the optimal performance under power limits is a challenging task and is commonly achieved through architecture and logic design, adjustments in the transistor/gate sizing, supply voltages or selection of the transistor thresholds. Solving this problem is challenging because it involves a hierarchical optimization over a number of discrete and continuous variables, with a combination of discrete and continuous constraints.

Various optimization techniques have been employed traditionally in digital circuit design, which range from simple heuristics to fully automated CAD tools. At circuit level, custom integrated circuits can be manually sized for minimum delay using the method of logical effort.[5] Technology mapping step in logic synthesis commonly employs delay minimization using gates with different sizes from a library of standard cells. TILOS[4] was the first tool that realized that the delay of logic gates expressed using Elmore's formula presents a convex optimization problem that can be efficiently minimized using geometric programming.[7] While the convex delay models used by TILOS are rather inaccurate because of their simplicity, the result is guaranteed to be *globally optimal.* Circuit delay optimization under constraints has been automated in the past as well. IBM's EinsTuner[3] uses a static timing formulation and tunes transistor sizes for minimal delay under total transistor width constraints. The delay models are obtained through simulation for better accuracy; however this guarantees only *local optimality.*

The conventional delay minimization techniques can be extended to account for energy as well. For example, a combination of both energy and delay, such as the energy-delay product (EDP) has been used as an objective function for minimization. A circuit designed to have the minimum EDP, however, may not be achieving the desired performance or could be exceeding the given energy budget. As a consequence, a number of alternate optimization metrics have been used that generally attempt to minimize an E^mD^n product.[1] By choosing parameters n and m a desired tradeoff between energy and delay can be achieved, but the result is difficult to propagate to higher layers of design abstraction. In the area of circuit design, this approach has been traditionally restricted to the evaluation of several different block topologies, rather than using it to drive the optimization.

In contrast, a systematic solution to this problem is to minimize the delay for a given energy constraint.[2] Note that a dual problem to this one, minimization of the energy

*Author to whom correspondence should be addressed.
Email: bora@eecs.berkeley.edu

subject to a delay constraint yields the same solution. Two solutions to this problem for sizing at circuit level are well known. The minimum energy of the fixed logic topology block corresponds to all devices being minimum sized. Similarly, the minimum delay point is well defined: At that point further upsizing of transistors yields no delay improvement.

Custom datapaths are an example of power-constrained designs where the designers traditionally iterate in sizing between schematics and layouts. The initial design is sized using wireload estimates and is iterated through the layout phase until a set delay goal is achieved. The sizing is refined manually using the updated wireload estimates. Finally, after minimizing the delay of critical paths, the non-critical paths are balanced to attempt to save some power, or in the case of domino logic to adjust the timing of fast paths. This is a tedious and often lengthy process that relies on the designer's experience and has no proof of achieving optimality. Furthermore, the optimal sizing depends on the chosen supply and transistor thresholds. An optimal design would be able to minimize the delay under power constraints by choosing supply and threshold voltages, gate sizes or individual transistor sizes, logic style (static, domino, pass-gate), block topology, degree of parallelism, pipeline depth, layout style, wire widths, etc.

This paper builds on the ideas of convex[4] or gradient-based[3] delay optimization techniques under constraints. The average energy per computation is used as a constraint for the delay minimization method. The ideas presented here constitute a modular design optimization framework for custom digital circuits in the power – performance space that:

- Formulates the design as a mathematical optimization problem;
- Uses a static timer to perform all circuit-related computations;
- Uses a mathematical optimizer to solve the optimization problem numerically;
- Adjusts various design variables at different levels of abstraction;
- Can employ different models in the timer in order to balance accuracy and convergence speed;
- Handles various logic families (static, dynamic, pass-gate) due to the flexibility of the modeling step;
- Guarantees the global optimality of the solution for certain families of analytical models that result in the optimization problem being convex;
- Verifies a near-optimality condition if global optimality cannot be guaranteed.

Section 2 describes the proposed design optimization framework. Section 3 discusses the models employed in framework and their tradeoffs. Section 4 presents results on two examples:

(1) A carry tree of a 64-bit adder in which sizing, supply, and threshold are tuned at the same time and

(2) a real-life application on 64-bit carry lookahead adders in the setup of a typical high performance microprocessor.

Finally, conclusions are presented in Section 5.

2. DESIGN OPTIMIZATION FRAMEWORK

The framework is built around a versatile optimization core consisting of a static timer in the loop of a mathematical optimizer, as shown in Figure 1.

The optimizer passes a set of specified design variables to the timer and gets the resulting cycle time (as a measure of performance) and power of the circuit, as well as other quantities of interest such as signal slopes, capacitive loads and, if needed, design variable gradients. The process is repeated until it converges to the optimal values of the design parameters that achieve the desired optimization goal. The circuit is defined using a SPICE-like netlist and the static timer employs user-specified models in order to compute delays, cycle times, power, signal slopes, etc. The choice of models depends on the tradeoffs between the desired accuracy and convergence speed and is discussed in Section 3.

Since the static timer is in the main speed-critical optimization loop, it is implemented in C++ to accelerate computation. It is based on the conventional longest path algorithm. The custom-written timer does not account for false paths or simultaneous arrivals, but it can be easily substituted with a more sophisticated one because of the modularity of the optimization framework.

The optimization core can be configured to perform various tasks for different types of circuits. For instance, if the circuit to be optimized is combinational, the framework can be configured to solve the following optimization problem:

Adjust GATE SIZES

in order to

Minimize DELAY

subject to:

Maximum ENERGY PER TRANSITION

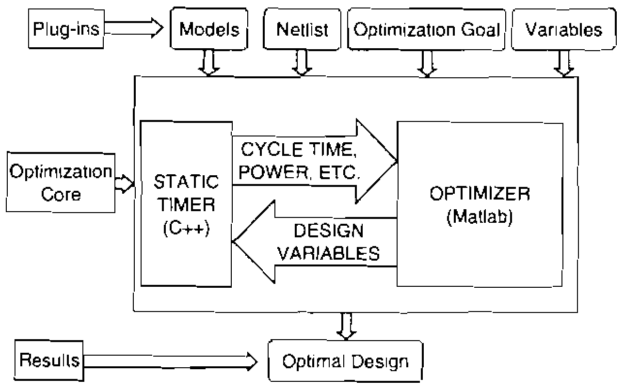

Fig. 1. Design optimization framework.

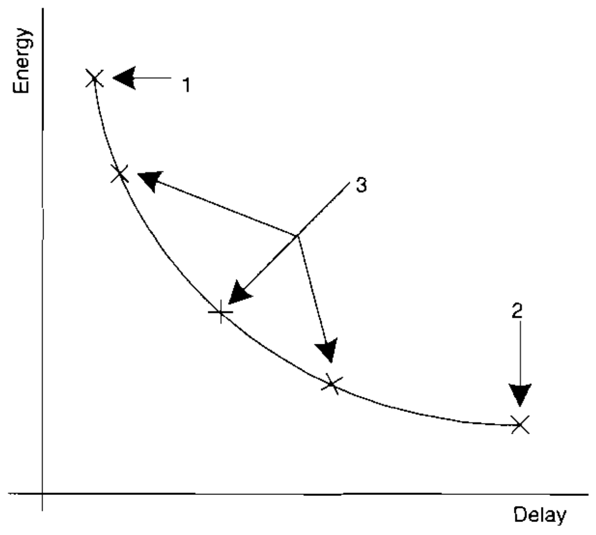

Fig. 2. Typical optimal energy–delay tradeoff curve for a combinational circuit.

with the following additional constraints (in order to ensure manufacturability and correct circuit operation):

Maximum internal slopes
Maximum output slopes
Maximum input capacitances
Minimum gate sizes

By solving this optimization problem for different values of the energy constraint, the optimal energy-delay tradeoff curve for that circuit is obtained, as shown in Figure 2.

The optimal tradeoff eurve has two well defined endpoints: Point 1 represents the fastest cireuit that can be designed; point 2 represents the circuit with the lowest energy per transition, primarily limited by minimum gate sizes and signal slope constraints. The points in-between the two extremes (marked "3" on the graph) correspond to minimizing various E^mD^n design goals (such as the EDP).

3. MODELS

Arbitrary optimization problems are very difficult to solve and the global optimality of the result cannot be usually guaranteed. If the functions involved in the optimization have certain mathematical properties, the problem becomes easier and certain statements can be made about the optimality of the results. In particular, convex optimization problems (where the objective and inequality constraint functions are convex[7]) can be solved reliably by commercial optimizers while guaranteeing tbe *global optimality* of the result.

For the circuit optimization framework from Figure 1, the properties of the objective and constraint functions are given by the models used in the static timer. Therefore, the choice of models in the static timer greatly influences

Table I. Comparison between analytical and tabulated models.

Analytical models	Tabulated models
– limited accuracy	+ very accurate
+ fast parameter extraction	– slow to generate
+ provide circuit operation insight	– no insight in the operation of the circuit
+ can exploit mathematical properties to formulate a convex optimization problem	– can't guarantee convexity; optimization is "blind"

the convergence speed and robustness of the optimizer. Analytical or tabulated models can be used in the optimization framework, depending on the desired accuracy and speed targets. Table I shows a comparison between the two main choices of models. Closed form analytical models can usually be forced into a convex form using various mathematical operations such as changes of variables and the introduction of additional (slack) variables.[7]

Tabulated models provide excellent accuracy at the points of characterization, but sacrifice the convexity property.

3.1. Analytical Models

In our initial optimizations we use a simple, yet fairly accurate analytical model. This model allows for a convex formulation of the resulting optimization problem, where the gate sizes are the optimization variables. The model has three components: A delay equation (1), a signal slope equation (2), and an energy equation (3):

$$t_D = p + g\frac{C_{\text{load}}}{C_{\text{in}}} + \eta \cdot t_{\text{slope\_in}} \quad (1)$$

$$t_{\text{slope\_out}} = \lambda + \mu\frac{C_{\text{load}}}{C_{\text{in}}} + \nu \cdot t_{\text{slope\_in}} \quad (2)$$

$$E = \sum_{\text{all\_nodes}} \alpha_i C_i V_{DD}^2 + T_{\text{cycle}} \sum_{\text{all\_gates}} W_j P_{\text{leak},j} \quad (3)$$

Equation (1) is an extension of the simple linear model used in the method of logical effort,[5] or the simplest model with limited accuracy used in commercial logie synthesis tools.[6] Equations (1) and (2) are a straightforward first order extension to these models that accounts for signal slopes.

The capaeitance of a node is computed using (4):

$$C_{\text{node}} = \sum_{\text{gate\_inputs\_at\_node}} k_i W_i + C_{\text{wire}} \quad (4)$$

where W_i are the corresponding gate sizes.

Each input of each gate is characterized for each transition by a set of seven parameters: p, g, η for the delay, λ, μ, ν for the slope and k for the capacitance. Each gate is also characterized by an average leakage power P_{leak} measured when its relative size is $W = 1$. Each node of the circuit has an activity factor α which is computed through logic simulation for a set of representative input patterns.

All the above equations can be written as posynomials in the gate sizes, W_i:

$$t_D = p + g\frac{\sum_{\text{driven\_gate\_inputs}} k_i W_i + C_{\text{wire}}}{kW_{\text{current}}} + \eta \cdot t_{\text{slope\_in}} \quad (5)$$

$$t_{\text{slope\_out}} = \lambda + \mu\frac{\sum_{\text{driven\_gate\_inputs}} k_i W_i + C_{\text{wire}}}{kW_{\text{current}}} + \nu \cdot t_{\text{slope\_in}} \quad (6)$$

If $t_{\text{slope\_in}}$ is a posynomial, then t_D and $t_{\text{slope\_out}}$ are also posynomials in W_i. By specifying fixed signal slopes at the primary inputs of the circuit, the resulting slopes and arrival times at all the nodes will also be posynomials in W_i. The maximum delay across all paths in the circuit will be the maximum of several posynomials, hence a generalized posynomial. A function f is a generalized posynomial if it can be formed using addition, multiplication, positive power, and maximum selection starting from posynomials.[7]

The energy equation is also a generalized posynomial: The first term is just a linear combination of the gate sizes while the second term is another linear combination of the gate sizes multiplied by the cycle time, that in turn is related to the delay through the critical path, hence also a generalized posynomial.

The optimization problem described in Section 2 using the above models has generalized posynomial objective and constraint functions:

Adjust W_i

in order to

$$\text{Minimize } \max(t_{\text{arrival, primary\_outputs}})$$

subject to:

$$E \le E_{\max},\ t_{\text{slope, primary outputs}} \le t_{\text{slope\_out, max}}$$

$$t_{\text{slope, internal nodes}} \le t_{\text{slope internal, max}}$$

$$C_{\text{primary inputs}} \le C_{\text{in, max}}$$

$$W_i \ge 1$$

Such an optimization problem with generalized posynomials is a *generalized geometric program* (GGP).[7] It can be converted to a convex optimization problem using a simple change of variables:

$$W_i = \exp(z_i) \quad (7)$$

With this change of variables the problem is tractable and can be easily and reliably solved by generic commercial optimizers. Moreover, since in convex optimization any local minimum is also global, the optimality of the result is *guaranteed.*

This delay model applies to any logic family where a gate can be represented through channel-connected components,[8] as in the case of complementary CMOS or domino logic. The limitation of this approach is that it uses linear approximations for the delay, signal slopes, and capacitances. Figure 3 shows a comparison of the actual and predicted delay for the rising transition of a gate for a fixed input slope and variable fanout. Since the actual delay is slightly concave in the fanout, the linear model is pessimistic at low and high fanouts and optimistic in the mid-range. The accuracy of the models can be increased by fitting them to higher order posynomials (hence maintaining the convexity of the optimization problem), but it results in exponentially increased time for characterization.

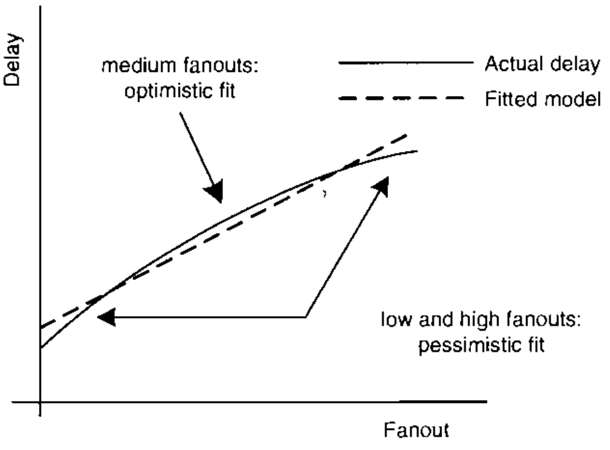

Fig. 3. Accuracy of fitted models.

3.2. Tabulated Models

If the accuracy of linear, analytical models is not satisfactory, tabulated models can be used instead. For instance, (1), (2) and their respective parameters can be replaced with the look-up table shown in Table II.

The table can have as many entries as needed for the desired accuracy and density of the characterization grid. Actual delays and slopes used in the optimization procedure are obtained through linear interpolation between the points in the table. The grid is non-uniform, with more points in the mid-range fanouts and slopes, where most designs are likely to operate. Additional columns can be added to the tables for different logic families—for instance if a dynamic gate is characterized this way, the relative size of the keeper to the pull-down network needs to be included, too.

The resulting optimization problem, even when using the change of variables from (7), cannot be proven to be convex. However, although not absolutely accurate, the

Table II. Example of a tabulated delay and slope model (NOR2 gate, input A, rising transition).

$C_{\text{load}}/C_{\text{in}}$	$t_{\text{slope\_in}}$	t_D	$t_{\text{slope\_out}}$
1	20 ps	19.3 ps	18.3 ps
…	…	…	…
10	200 ps	229.6 ps	339.8 ps

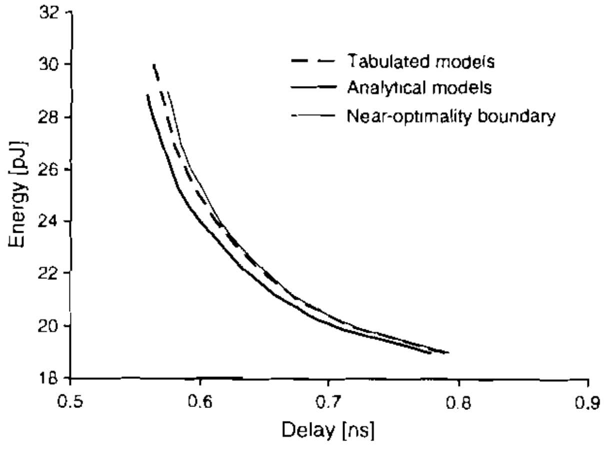

Fig. 4. Analytical versus tabulated models and near-optimality boundary.

analytical models that describe the behavior of the circuits closely approximate the tabulated models. Thus, the resulting optimization problem is nearly-convex and can still be solved with very good accuracy and reliability by the same optimizers as before.[9] The result of the nearly-convex problem can be checked against a near-optimality boundary. The example in Figure 4 shows a comparison of the analytical and tabulated models and the corresponding near-optimality boundary.

The figure shows the energy-delay tradeoff curves for an example 64-bit Kogge-Stone carry tree in static CMOS using a 130 nm process. The same circuit is optimized using each of the two model choices discussed in this section. Both models show that the fastest static 64-bit carry tree can achieve the delay of approx. 560 ps, while the lowest achievable energy is 19 pJ per transition. The analytical models are slightly optimistic because the optimal designs exhibit mid-range gate fanouts where the analytical models tend to underestimate the delays (Fig. 3).

The near optimality boundary is obtained by using tabulated models to compute the delay and energy of the designs that resulted from the optimization with analytical models. This curve represents a set of designs optimized using analytical models, but evaluated with tabulated models. Since those designs are guaranteed to be optimal for analytical models, the boundary is within those models' error of the actual global optimum. However, if an optimization using the correct models (tabulated) converges to the correct solution, it will always yield a better result than a re-evaluation of the results of a different optimization using the same models. Therefore, if the optimization with tabulated models is to converge correctly the result must be within the near-optimality boundary i.e., will have a smaller delay for the same energy.

If a solution obtained using tabulated models is within the near-optimality boundary it will be deemed "near-optimal" and hence acceptable.

In a more general interpretation, optimizing using tabulated models is equivalent to optimizing using a trusted timing signoff tool whose main feature is very good accuracy. The result of such an optimization is not guaranteed to be globally optimal. The near-optimality boundary is obtained by running the timing signoff tool on a design obtained from an optimization that can guarantee the global optimality of the solution. The comparison is fair because the power and performance figures on both curves are evaluated using the same (trusted and accurate) timing signoff tool.

3.3. Model Generation and Accuracy

Tabulated models are generated through simulation. The gate to be modeled is placed in a simple test circuit and the fanout and input slope and relative keeper size (for dynamic gates) are adjusted using automated Perl scripts. The simulator is invoked iteratively for all the points in the table and the relevant output data (delay, output slope) is stored. This can be lengthy (although parallelizable) if the grid is very fine and the number of points large. This characterization is similar to the one performed for the standard-cell libraries, and yields satisfactory accuracy. Alternatively, or in addition, characterization points for static gates can be used from the tabulated entries in the standard cell library.

Analytical models are obtained through data fitting. Data points are obtained through simulation in the same manner as for tabulated models. Least squares fitting is used to obtain the parameters of the models. The number of points required for a good fit (50–100, depending on the model) is less than the number of points needed for tabulated models (of the order of 1000) and thus the characterization time for analytical models is one order of magnitude shorter.

The error of the analytical models depends on their complexity and on the desired data range. The models in (1) and (2) are accurate within 10% of the actual (simulated) delays and slopes for the range specified in Table II. The energy equation (3) is accurate within 5% for fast slopes but its accuracy degrades to 12% underestimation at slow input slopes due to the crowbar current (which is not included in the equation). The maximum slope constraints for output and internal nodes ensure such worst cases do not occur in usual designs.

4. RESULTS

We use the presented optimization framework to optimize a 64-bit adder, which is a very common component of custom datapaths. The critical path of the adder consists of the carry computation tree and the sum select.[10] Tradeoffs between the performance and power can be performed through the selection of circuit style, logic design of carry equations, selection of a tree that calculates the carries, as well as through sizing and choices of supply voltages and transistor thresholds.

Carry-lookahead adders are frequently used in high-performance microprocessor datapaths. Although adder design is a well-documented research area,[11–14] fundamental understanding of their energy-delay performance at the circuit level is still largely invisible to the microarchitects. The optimization framework presented in this paper provides a means of finding the energy budget breakpoint where the architects should change the underlying circuit design.

Datapath adders are good example for the optimization because their layout is often bit-sliced. Therefore, the ~~critical wire lengths can be estimated pre-design and~~ are a weak function of gate sizing. The optimization is performed on two examples:

(1) A 64-bit carry tree of a carry-lookahead adder implemented in standard static CMOS, using analytical models to tune gate sizes, supply, and threshold voltages;

(2) 64-bit carry lookahead adders implemented in domino and static CMOS, using tabulated models.

4.1. Tuning Sizes, Supply, and Threshold Using Analytical Models

In order to tune, supply, and threshold voltages, the models must include their dependencies. A gate equivalent resistance can be computed from analytical saturation current models (a reduced form of the BSIM3v3[15, 16]):

$$R_{EQ} = \frac{1}{V_{DD}/2}\int_{V_{DD}/2}^{V_{DD}} \frac{V_{DS}dV_{DS}}{I_{DSAT}} = \frac{3}{4}\frac{V_{DD}(\beta_1 V_{DD}+\beta_0+V_{DD}-V_{TH})}{W\cdot K(V_{DD}-V_{TH})^2}\left(1-\frac{7V_{DD}}{9V_A}\right) \quad (8)$$

Using (8), supply and threshold dependencies can be included in the delay model. For instance (1) becomes (9), with (2) having a very similar expression:

$$t_D = c_2 R_{EQ} + c_1 R_{EQ}\frac{C_{\text{load}}}{C_{\text{in}}} + (\eta_0+\eta_1 V_{DD})\cdot t_{\text{slope\_in}} \quad (9)$$

The model is accurate within 8% of the actual (simulated) delays and slopes around nominal supply and threshold, over a reasonable yet limited range of fanouts (2.5–6). For a ±30% range in supply and threshold voltages the accuracy is 15%.

Figure 5 shows the optimal energy-delay tradeoff curves of a 64-bit Kogge-Stone carry tree implemented in static CMOS in three cases:

(1) Only gate sizes are optimized for various fixed supplies and the nominal threshold;

(2) Gate sizes and supply are optimized for nominal threshold;

(3) Gate sizes, supply, and threshold voltage are optimized jointly.

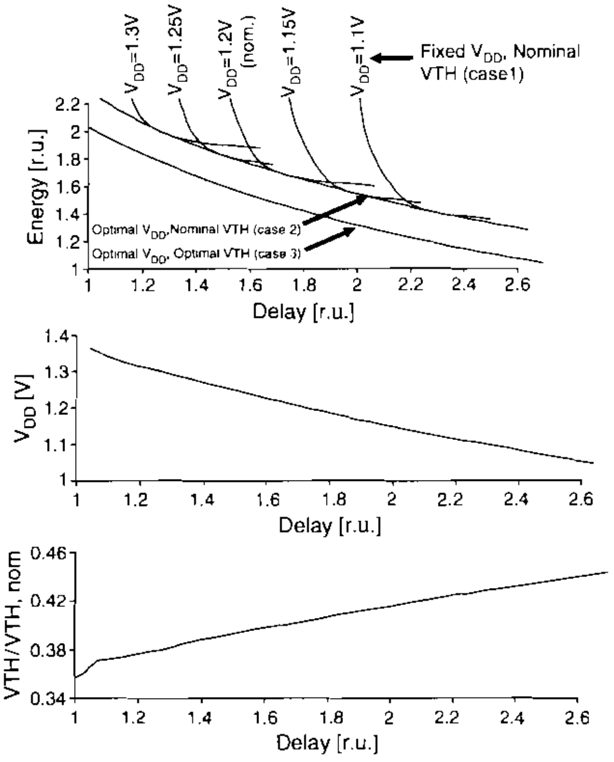

Fig. 5. Energy–delay tradeoff curves for different sets of optimization variables and corresponding supply and threshold voltage.

Figure 5 also shows the corresponding optimal supply voltage for case 2 and the corresponding optimal threshold for case 3 normalized to the nominal threshold voltage of the technology.

A few interesting conclusions can be drawn from the above figures:

- The nominal supply voltage is optimal in exactly one point, where the $V_{DD} = 1.2$ V curve is tangent to the optimal V_{DD} curve. In that point, the sensitivities of the design to both supply and sizing are equal;[2]
- Power can be reduced by increasing V_{DD} and downsizing if the V_{DD} sensitivity is less than the sizing sensitivity;
- Achieving the last few picoseconds of the delay reduction is very expensive in energy because of the large sizing sensitivity (curves are very steep at low delays);
- The optimal threshold is well below the nominal threshold. For such a high activity circuit, the power lost through increased leakage is recuperated by the downsizing afforded by the faster transistors with lower threshold. Markovic et al.,[2] came to a similar conclusion using an analytical approach.

4.2. Tuning Sizes in 64-bit CLA Adders Using Tabulated Models

Using tabulated models as described in Section 3, various adder topologies implemented in different logic families are optimized in the energy–delay space under the typical loading for a microprocessor datapath. Details about

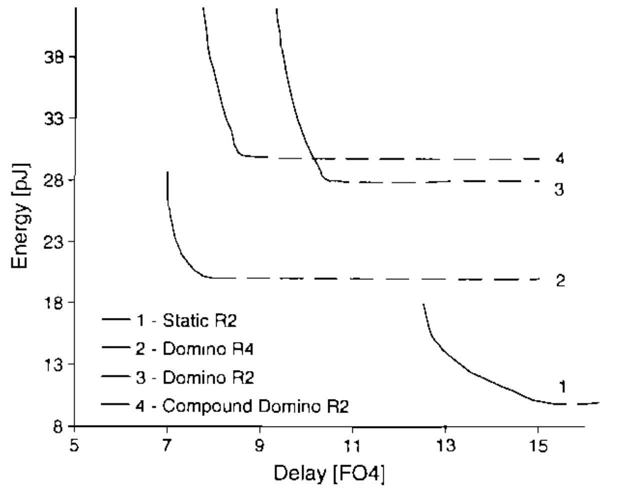

Fig. 6. Energy–delay tradeoff curves for selected 64-bit CLA adders.

the logic structure of the adders can be found in (Ref. [17]). Figure 6 shows the energy–delay tradeoff curves for a few representative adder configurations in a general-purpose 130 nm process. Radix-2 (R2) adders merge 2 carries at each node of the carry tree. For 64 bits, the tree has 6 stages of relatively simple gates. Radix-4 (R4) adders merge 4 carries at each stage, and therefore a 64-bit tree has only 3 stages but the gates are more complex. In the notation used in Figure 8 classical domino adders use only (skewed) inverters after a dynamic gate, whereas compound domino use more complex static gates, performing actual radix-2 carry-merge operations.[18]

Based on these tradeoff curves, microarchitects can clearly determine that under these loading conditions radix-4 domino adders are always preferred to radix-2 domino adders. For delays longer than 12.5 FO4 inverter delays, a static adder is the preferred choice because of its lower energy.

The fastest adder implements Ling's pseudo-carry equations in a domino radix-4 tree with a sparseness factor of 2.[17] An implementation of the fastest adder in a general-purpose 90 nm process is described in (Ref. [19]) and measured results are in good agreement with the optimizer.

4.3. Runtime Analysis

The complexity and runtime of the framework depend on the size of the circuit. Small circuits are optimized almost instantaneously. A 64-bit domino adder with 1344 gates (a fairly large combinational block) is optimized on a 900 MHz P3 notebook computer with 256 MB of RAM in 30 seconds to 1 minute if the constraints are rather lax. When the constraints are particularly tight and the optimizer struggles to keep the optimization problem feasible, the time increases to about 3 minutes. A full power – performance tradeoff curve with 100 points can be obtained in about 90 minutes on such a machine. For grossly infeasible problems the optimizer provides a "certificate of infeasibility" in a matter of seconds.

For large designs the framework allows gate grouping. By keeping the same relative aspect ratio for certain groups of gates, the number of variables can be reduced and the runtime kept reasonable. Gate grouping is a natural solution for circuits with regular structure. For instance, in an adder, gates can be grouped at various levels of the carry tree, which simplifies the layout. All the adders optimized in Section 4.1 and 4.2 use gate grouping for identical gates in the same stage.

5. CONCLUSIONS

This paper presents a design optimization framework that tunes custom digital circuits based on a static timing formulation. The framework can use a wide variety of models and tune different design variables. The problem solved is generally an energy-constrained delay minimization. Due to the flexibility in choosing models, the framework can easily handle various logic families.

If analytical models are used the optimization is convex, can be easily and reliably solved, and its results are guaranteed to be optimal. The accuracy of the modelling can be improved by using look-up tables, at the cost of the optimality guarantee as well as increased characterization time and complexity. More generally, the optimization can be run on any trusted and accurate timing signoff tool, with the same tradeoffs and limitations as for tabulated models. Results obtained using tabulated models (or with the said "trusted and accurate timing signoff tool") can be verified against a near-optimality boundary computed from results guaranteed optimal in their class. If the results fall within that boundary they are considered near-optimal and therefore acceptable.

The framework was demonstrated on 64-bit carry-lookahead adders in 130 nm CMOS. A static Kogge-Stone tree was tuned using analytical models by adjusting gate sizes, supply voltage, and threshold voltage. Complete domino and static 64-bit adders were also tuned in a typical high performance microprocessor environment using tabulated models by adjusting gate sizes.

The framework can be extended to optimize sequential blocks as well. One aspect of this optimization could involve the placement of the latch positions in a pipelined datapath. By building on the combinational circuit optimization, this tool would allow microarchitects a larger freedom in trading off cycle time for latency. Another interesting extension of this framework is to optimize the energy-delay of a block under the presence of uncertainty. The convex delay models can be extended to include the parameter uncertainty due to process or environment variations. By using these models, the GGP translates into a robust GP.[20]

Acknowledgment: This work was supported in part by NSF grant ECS-0238572.

References

1. P. I. Penzes and A. J. Martin, Energy–delay efficiency of VLSI computations. *Proceedings of the 14th Great Lakes Symposium on VLSI* **(2002)**, pp. 104–111.
2. D. Markovic, V. Stojanovic, B. Nikolić, M. Horowitz, and R. W. Brodersen, Methods for true energy-performance optimization. *IEEE J. Solid State Circuits* **(2004)**, Vol. 39, pp. 1282–1293.
3. A. R. Conn, I. M. Elfadel, W. W. Molzen, Jr., P. R. O'Brien, P. N. Strenski, C. Visweswariah, and C. B. Whan, Gradient-based optimization of custom circuits using a static timing formulation. *Proceedings of 36th Design Automation Conference* **(1999)**, pp. 452–459.
4. J. P. Fishburn and A. E. Dunlop, TILOS: A posynomial programming approach to transistor sizing. *Proceedings of IEEE International Conference on Computer-Aided Design* **(1985)**, pp. 326–328.
5. I. Sutherland, R. Sproul, and D. Harris, *Logical Effort*, Morgan-Kaufmann **(1999)**.
6. Synopsys. Synopsys® Design Compiler User's Manual Version 2004.12 **(2004)**.
7. S. Boyd and L. Vandenberghe, Convex Optimization, Cambridge University Press **(2003)**.
8. R. Zlatanovici, Master Thesis, UC Berkeley **(2002)**.
9. Mathworks. Matlab® Optimization Toolbox User's Guide Version 3 **(2004)**.
10. J. M. Rabaey, A. Chandrakasan, and B. Nikolić, Digital Integrated Circuits: A Design Perspective, 2nd edn., Prentice-Hall **(2003)**.
11. P. M. Kogge and H. S. Stone, A parallel algorithm for efficient solution of a general class of recursive equations. *IEEE Transactions on Computers* **(1973)**, Vol. 22, pp. 786–793.
12. J. Park, H. C. Ngo, J. A. Silberman, and S. H. Dhong, 470 ps 64 bit parallel binary adder. *Proceedings of Symposium on VLSI Circuits* **(2000)**, pp. 192–193.
13. T. Han and D. A. Carlson, Fast area efficient VLSI adders. *8th Symposium on Computer Arithmetic* **(1987)**, pp. 49–56.
14. S. Naffziger, A sub-nanosecond 0.5 μm 64b adder design. *Proceedings of International Solid-State Circuits Conference* **(1996)**, pp. 210–211.
15. K. Y. Toh, P. K. Ko, and R. G. Meyer, An engineering model for short-channel CMOS devices. *IEEE J. Solid State Circuits* **(1998)**, Vol. 23, pp. 950–958.
16. J. Garrett, Master Thesis, UC Berkeley **(2004)**.
17. R. Zlatanovici and B. Nikolić, Power – performance optimal 64-bit carry-lookahead adders. *Proceedings of European Solid State Circuit Conference* **(2003)**, pp. 321–324.
18. H. Q. Dao, B. R. Zeydel, and V. G. Oklobdzija, Energy minimization method for optimal energy–delay extraction. *Proceedings of European Solid State Circuit Conference* **(2003)**, pp. 177–180.
19. S. Kao, R. Zlatanovici, and B. Nikolić. A 240 ps 64 b carry-lookahead adder in 90 nm CMOS. *International Solid-State Circuits Conference* **(2006)**, pp. 438–439.
20. D. Patil, S. Yun, S. Kim, A. Cheung, M. Horowitz, and S. Boyd, A new method for design of robust digital circuits. *International Symposium on Quality Electronic Design* **(2005)**, pp. 676–681.

Radu Zlatanovici

Radu Zlatanovici *received his B.S. and M.S. degrees from Politehnica University Bucharest, Romania in 1999 and 2000 respectively. In 2000 he joined the University of California at Berkeley where he received an M.S. degree in 2002. He is currently working toward his Ph.D. degree at the same university. He was on the faculty of Politehnica University Bucharest from 1999 to 2000. In 2002 and 2003 he interned at IBM T. J. Watson Research Center, Yorktown Heights, NY working on power – performance tradeoffs for pipelined digital circuits. His research activities include optimization of digital circuits in the power – performance space, high-speed and low-power arithmetic circuits, and design of digital circuits using alternative devices such as FinFETs.*

Borivoje Nikolić

Borivoje Nikolić *received the Dipl.Ing. and M.Sc. degrees in electrical engineering from the University of Belgrade, Yugoslavia, in 1992 and 1994, respectively, and the Ph.D. degree from the University of California at Davis in 1999. He was on the faculty of the University of Belgrade from 1992 to 1996. He spent two years with Silicon Systems, Inc., Texas Instruments Storage Products Group, San Jose, CA, working on disk-drive signal processing electronics. In 1999, he joined the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley where he is now an Associate Professor. His research activities include high-speed and low-power digital integrated circuits and VLSI implementation of communications and signal-processing algorithms. He is coauthor of the book Digital Integrated Circuits: A Design Perspective, 2nd edn. (Prentice-Hall, 2003). Dr. Nikolić received the NSF CAREER award in 2003, College of Engineering Best Doctoral Dissertation Prize and Anil K. Jain Prize for the Best Doctoral Dissertation in Electrical and Computer Engineering at University of California at Davis in 1999, as well as the City of Belgrade Award for the Best Diploma Thesis in 1992.*