

---

# Energy-Performance Optimization of Synthesized Digital Integrated Circuits

by Seng Oon Toh

---

## Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences,  
University of California at Berkeley, in partial satisfaction of the requirements for the  
degree of **Master of Science, Plan II**

Approval for the Report and Comprehensive Examination

Committee:

---

Professor Borivoje Nikolic  
Research Advisor

---

(Date)

\*\*\*\*\*

---

Professor Elad Alon

---

(Date)

# Contents

Contents .....	i
List of Figures .....	iii
List of Tables .....	iv
1.0 Introduction .....	1
1.1 Background and current state of the art .....	1
1.2 Thesis overview.....	4
1.3 Thesis organization .....	5
2.0 Standard Cell Library and Gate Delay Models .....	6
2.1 Synopsys Non-linear Delay Model (NLDM).....	7
2.2 Posynomial Delay Model.....	8
2.3 Accuracy of Posynomial Gate Delay Models.....	10
3.0 Combinational Circuit Optimization .....	14
3.1 Static Timer Geometric Program.....	14
3.2 Optimization Using Global Fit Parameters .....	18
3.3 Local Fit Parameters.....	20
3.4 Optimization Using Local Fit Parameters.....	23
3.5 Localized Convex Fit of Concave Gate Delay Models .....	26
4.0 Snapping Back Continuous Gate Sizes to Discrete Gate Sizes.....	34
4.1 Simple Rounding Algorithms.....	34
4.2 Round and Resize Algorithm .....	35
4.3 Continuous Solution Guided Dynamic Programming.....	37
4.4 Continuous Solution Guided Greedy Sizing.....	38
5.0 Optimizing Sequential Circuits .....	43
5.1 Flip-flop Gate Model .....	43
5.2 Optimization of Circuits with Sequential Elements.....	47
5.3 Efficiently Optimizing Large Sequential Circuits .....	50
6.0 Case Study : Optimizing Building Blocks of a Double-Precision Floating Point Unit 55	
6.1 Double-Precision Floating Point Unit .....	55
6.2 Optimal Selection of Process Technology .....	57
6.3 53-Bit Multiplier .....	59

6.4	108-Bit Adder.....	61
6.5	55-Bit Incrementer .....	62
6.6	Exponent Datapath.....	64
6.7	Combining Multiple Blocks.....	65
7.0	Conclusion.....	75
7.1	Future Work.....	76
	Bibliography .....	78

## List of Figures

Figure 2-1. Illustration of Synopsys NLDM interpolation. ....	7
Figure 2-2. Accuracy of single-gate-fit posynomial gate delay model compared to tabulated values.....	10
Figure 2-3. Accuracy of multiple-gate-fit posynomial gate delay model compared to tabulated values.....	11
Figure 3-1. Static timer delay constraints for 2-input NAND gate.....	15
Figure 3-2. Interconnect delay model.....	19
Figure 3-3. Local fit iterative optimization algorithm.....	24
Figure 3-4. Progress of iterative local-fit optimization in energy-delay space.....	26
Figure 3-5. Convergence of iterative local fit delay minimization.....	27
Figure 3-6. Difference in gate sizes between successive delay minimization runs.....	28
Figure 3-7. Convergence of iterative local fit constrained delay energy minimization....	29
Figure 3-8. Difference in gate sizes for two successive energy minimization runs.....	30
Figure 3-9. Local posynomial fitting of gate delay with gate sizing.....	30
Figure 4-1. Greedy sizing algorithm.....	38
Figure 4-2. Evaluation of change in slack due to upsizing of a gate.....	39
Figure 4-3. Results of rounding using greedy sizing algorithm.....	40
Figure 5-1. Schematic of a D flip-flop.....	44
Figure 5-2. Clock-output delay of flip-flops with different drive strengths.....	45
Figure 5-3. Setup time of flip-flops with different drive strengths.....	46
Figure 5-4. Computation time versus number of gates in the digital circuit.....	51
Figure 6-1. Block diagram of the mantissa datapath in a double-precision FPU.....	56
Figure 6-2. Energy-delay tradeoff curves for a 16-bit multiplier at different threshold voltages.....	58
Figure 6-3. Optimization results for 53-bit multiplier.....	60
Figure 6-4. Optimization results for 108-bit adder.....	61
Figure 6-5. Optimization Results for 55-Bit Incrementer.....	63
Figure 6-6. Verilog description of exponent datapath.....	64
Figure 6-7. Optimization results for exponent datapath.....	65
Figure 6-8. Energy-delay tradeoff curves for 161-bit leading zero detector with different $C_{in,max}$ constraints.....	67
Figure 6-9. Energy-delay tradeoff curves for 161-bit leading zero detector with different $C_{load}$ constraints.....	67
Figure 6-10. Energy-delay composition of two blocks connected in series.....	69
Figure 6-11. Composite energy-delay curves of all possible combinations of one's complement block followed by LZD block.....	70
Figure 6-12. Normalization and rounding logic of a double-precision FPU.....	71
Figure 6-13. Comparison between composite curve and synthesis design-points of FPU normalization-rounding logic.....	72
Figure 6-14. Illustration of capacitance constraints placed on inputs and outputs of circuit blocks.....	73

## List of Tables

Table 3-1: Results of circuit optimization using global models. ....	19
Table 3-2: Gate delay lookup table for actual gate with size 1 (data normalized to delay of 2X gate with 0.0031 ns input slope and 1 unit normalized output capacitance). ....	21
Table 3-3: Gate delay lookup table for actual gate with size 2 (data normalized to delay of 2X gate with 0.0031 ns input slope and 1 unit normalized output capacitance). ....	21
Table 3-4: Evaluation of optimization results using local fit parameters. ....	22
Table 5-1: Computation time for optimizing a 53-bit multiplier. ....	52
Table 5-2: Runtime required for optimization of complete double-precision FPU Circuit and separate pipeline stages. ....	54

## 1.0 Introduction

Synthesized standard-cell digital integrated circuits are commonly used as a vehicle for implementation of application specific integrated circuits (ASICs) and many commercial tools are available to accomplish this purpose. Designs are usually synthesized using a sequential method where timing constraints are first met before optimizing energy and area as secondary objectives. This methodology often results in designs that are suboptimal in the secondary objectives. We propose a convex optimization framework that is able to jointly optimize energy and performance in a standard cell design to achieve the global optimum under relaxation of discrete sizing constraints. The optimizer is used post-synthesis on a netlist of a mapped design to select optimal gate sizes and is then tested on a synthesized 16-bit multiplier and a pipelined double-precision floating point unit.

### 1.1 Background and current state of the art

Energy-performance optimization of digital integrated circuits has been used for quite some time [Ruehli77] and has developed into three major forms: heuristic, analytical, and hybrid. Heuristic methods such as TILOS [Fishburn85] and [Lin90] generally perform this optimization by iteratively optimizing the critical path while employing a heuristic to resize gates along the path, terminating when there are no more candidates. Unfortunately, near-critical paths do affect the timing of the critical path if they are in the fanout of that path. The omission of these paths in the greedy gate sizing selection results in sub-optimal results. [Coudert96] considers all paths in the

optimization by performing a multi-dimension descent-based optimization employing a heuristic to limit the computational complexity of gradient recomputation. [Chinnery05] formulates the gate sizing problem as a linear program (LP) with cell-choice variables  $\gamma_v \in [0,1]$ , for each gate  $v$ , which specifies the affinity of the gate choice between the current gate size and the next best gate size. The LP is then solved and the next best gate is used if  $\gamma_v$  is greater than 0.99. [Nguyen03] and [Orshansky05] perform energy minimization by jointly assigning threshold voltages and gate sizes in an iterative approach that optimizes slack distribution among the gates followed by exhaustive local search for implementations that maximize energy savings. While these methods work directly with the discrete standard cell gate sizes and can employ highly accurate timing models, they optimize delay and energy separately in sequential steps instead of considering both energy and performance concurrently. Since the quality of results depends on starting values, these algorithms are susceptible to being trapped in local minima and are unable to guarantee global optimality.

Realizing that the energy-performance optimization of discrete-sized gates is NP-complete [Chan90], researchers set out to produce better analytical solutions by relaxing the constraint of discrete sizes. This relaxation allows the formulation of the optimization in a form that can be solved efficiently, producing a result that is as accurate as the analytical models used, such as in [Conn99]. Furthermore, recognizing that convex models such as the Elmore delay model used in TILOS can be solved exactly using convex optimization; there has been renewed interest in modeling this optimization problem as a geometric program (GP) which can be solved for global optimum such as in [Boyd05] and [Zlatanovici06]. This interest is fueled by the recent availability of fast

interior-point method based optimizers such as [Mosek07] which provide the user with flexibility in formulating the digital circuit optimization separately and relying on a general purpose solver to yield a globally optimum solution. These geometric programs are robust enough to account for slope, wire parasitic, rise/fall transitions, and rail voltage dependence. [Chinnery06] extends the convex optimization of custom circuits to ASICs by fitting the posynomial gate delay and power models to data generated from standard cells. Unfortunately, no matter how much effort is placed on accurately modeling the parameters of the gates, the non-convexity of these parameters ultimately result in fitting errors which degrade the quality of the results obtained. These analytical methods also assume that a good discrete solution can be obtained from the continuous solution through a simple rounding process, but this can not be guaranteed primarily due to the potential sparseness of gate sizes available from the standard cell library.

Hybrid methods try to recover the loss in optimality from the rounding process by applying non-linear optimization or heuristics on the analytical solution. [Chuang95] solves the analytical optimization problem as a linear program and uses a branch-and-bound algorithm to move from the continuous solution to a discrete solution. Recently [Hu07] demonstrated a hybrid method which first solves a geometric program of the relaxed optimization problem and performs dynamic programming to arrive at a discrete solution using the analytical solution as a guide for limiting the search space. The authors use an Elmore delay model for both optimization steps which is known to produce gross mismatch between actual delays. Although this technique shows great promise, further work needs to be done to examine the effectiveness of this algorithm under more accurate gate delay models.

A robust energy-performance optimization tool for standard cell designs should be able to produce a global optimum solution while considering the complicated interactions between energy, performance, sizing, supply, threshold, and process corners. To this end, the geometric program analytical method is the best candidate but only produces a continuous solution which also suffers from inaccuracies due to the need for convex models. A hybrid method that leverages the utility of a geometric program formulation while using heuristics to arrive at a discrete solution using accurate models therefore shows the greatest promise in solving the energy-performance optimization problem effectively.

## 1.2 Thesis overview

This thesis inherits the existing optimization framework developed by [Zlatanovici06]. This previous work formulates the energy-performance optimization problem as a convex optimization problem with a globally optimum solution. This thesis further develops this existing framework with the goals of:

- Constraining the choices of gates to only those available in a standard cell library;
- Adapting non-linear concave gate delay models to linear localized models that allow solving for the global optimum;
- Developing techniques for converging to a good solution with concave gate delay models;
- Investigating properties of pipelined circuits that can be utilized to reduce computation time of the optimizer.

### 1.3 Thesis organization

Chapter 2 introduces the concept of a standard cell library as well as the gate delay models that are used for commercial static timing analysis. A commonly used linear approximation of these models will then be described followed by a detailed analysis on the shortcomings of this model when applied to actual standard cell libraries. Chapter 3 describes an algorithm for fixing the problem of inaccurate global gate delay models by iteratively solving the gate sizing geometric program using more accurate localized models. The results of applying this algorithm to a 16-bit synthesized multiplier will be analyzed and a heuristic will be introduced to improve convergence of the algorithm. Chapter 4 provides a survey of techniques used for snapping back continuous gate sizes to discrete gate sizes available in the standard cell library. A simple greedy rounding algorithm will be applied on a continuously sized netlist and the resulting energy and performance numbers will be compared with that of a commercial tool. Chapter 5 investigates properties of digital circuits that can be exploited to efficiently solve the GP. Chapter 6 analyzes the quality of the results produced by the optimizer by applying the tool on various blocks in a floating point unit. This chapter also analyzes the tradeoffs involved in selecting the best process options from a process technology for a given design. A heuristic for combining the energy-delay tradeoff curves of individual blocks is also presented and tested. Finally, conclusions and suggestions for future research are presented in Chapter 7.

## 2.0 Standard Cell Library and Gate Delay Models

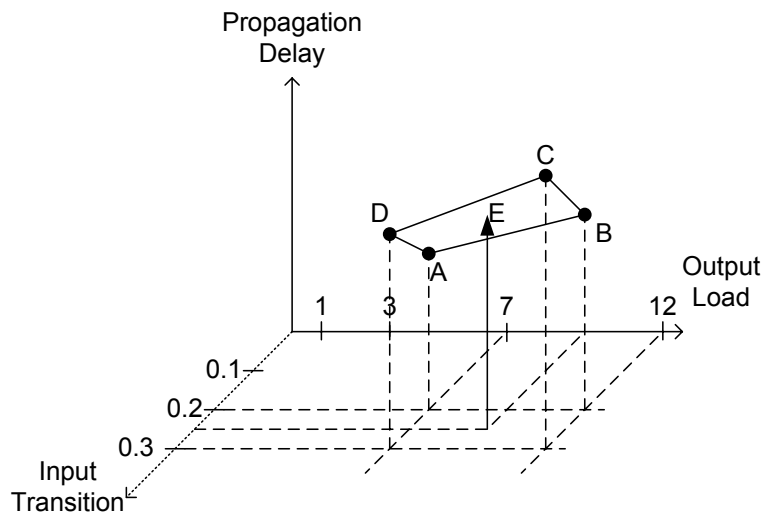
A standard cell library is a set of transistors which implement fixed logic functions such as INVERT, NAND, NOR, and XOR or storage functions such as D flip-flops or latches. They are usually available with corresponding abstract layout in Library Exchange Format (LEF), schematic, and register transfer level (RTL) views which allow circuit designers to synthesize, place, and route digital integrated circuits at a high level. During the process of synthesis, logic described in a RTL language is transformed into a canonical Boolean representation which is used to guide the selection of standard cell gates to implement the logic. Physical constraints such as maximum input capacitance, output load capacitance, and maximum delay are often placed on these designs.

In order to meet these constraints, standard cell libraries provide implementation of each logic function with different drive strengths. For example there might be an INVERT cell available with 1X, 2X, and 4X drive strengths. The 2X INVERT cell generally has twice the drive current than that of the 1X cell. Ignoring the delay associated with self-loading capacitance of the gates, a 2X INVERT cell can switch a capacitive load at half the time required for a 1X INVERT cell.

The process of selecting gate sizes is called “gate sizing.” Standard cell libraries only make available gates in discrete sizes that follow an arithmetic or a geometric series. There are also standard cell libraries that automatically generate cells based on required drive strengths, which effectively provides a standard cell library with continuous gate sizes such as [Hashimoto04].

## 2.1 Synopsys Non-linear Delay Model (NLDM)

The commonly used commercial gate delay model is Synopsys Non-linear Delay Model (NLDM), which is a two-dimensional lookup table for gate delay and output transition slope indexed by output load and input slope. Although each gate is usually characterized using a similar set of input slopes, the gates are characterized using a different set of output loads depending on gate sizes. For example, a 1X INVERT gate would be characterized at smaller output load capacitances versus a 4X INVERT gate because the larger gate is meant to drive larger loads. Data points corresponding to input slope and output load pairs that are not on the grid are interpolated based on the closest 2x2 data points. Figure 2-1 illustrates the interpolation process for a hypothetical gate.



[Bhatnagar02]

Figure 2-1. Illustration of Synopsys NLDM interpolation.

To calculate the delay of the a hypothetical gate at an input slope of 0.2 and an output load of 10, points A, B, C, and D are used to generate the coefficients ( $p$ ,  $g$ ,  $\eta$ ,  $\zeta$ ) for the following equation:

$$t_D = p + g \cdot C_L + \eta \cdot t_{slope,in} + \zeta \cdot C_L \cdot t_{slope,in} \quad (2-1)$$

This equation models exactly the surface illustrated in Figure 2-1. The four coefficients can be solved by performing a least-squares fit of the equation based on tabulated values from the four closest points. Coefficient  $p$  models the intrinsic delay of the gate while coefficient  $g$  models the logical effort of the gate. Coefficient  $\eta$  models the dependence of gate delay on input slope, while the last coefficient  $\zeta$  models the interaction between input slope and output load on delay. The value at point E is then calculated using (2-1). Note that although this equation (minus the interaction term) looks exactly like Equation (3-13) in [Zlatanovici06], these coefficients are only valid for input slopes and output loads within the points A, B, C, and D. These parameters can be considered as localized parameters of the gate. The same interpolation method also applies for the calculation of output slope except that the Z-axis is replaced by the output slope.

## 2.2 Posynomial Delay Model

The following section describes the gate delay model that is employed in the optimization framework described by [Zlatanovici06]. The reader is referred to the

reference for a more detailed coverage of this model. Gate delays are modeled using the following two equations which describe the propagation delay as well as the slope at the output of a gate. Each gate in the standard cell library is described using 6 parameters,  $p$ ,  $g$ ,  $\eta$ ,  $\lambda$ ,  $\mu$ , and  $\nu$ .

$$t_D = p + g \cdot \frac{C_L}{C_{in}} + \eta \cdot t_{slope,in} \quad (2-2)$$

$$t_{slope,out} = \lambda + \mu \cdot \frac{C_L}{C_{in}} + \nu \cdot t_{slope,in} \quad (2-3)$$

According to [Zlatanovici06], these equations can be converted into the following posynomials with gate sizes ( $W_i$ ) as variables.

$$t_D = p + g \cdot \frac{\sum K_i \cdot W_i}{K_{current} \cdot W_{current}} + \eta \cdot t_{slope,in} \quad (2-4)$$

$$t_{slope,out} = \lambda + \mu \cdot \frac{\sum K_i \cdot W_i}{K_{current} \cdot W_{current}} + \nu \cdot t_{slope,in} \quad (2-5)$$

The variables  $K_i$  in the equation represent the unit-sized input capacitance of the gates. Since  $t_{slope,in}$  can be recursively expanded out as posynomials in  $W_i$ , the equations

are thus posynomials which can be formulated as a geometric program. The only limitation of this formulation is that all the coefficients have to be non-negative.

### 2.3 Accuracy of Posynomial Gate Delay Models

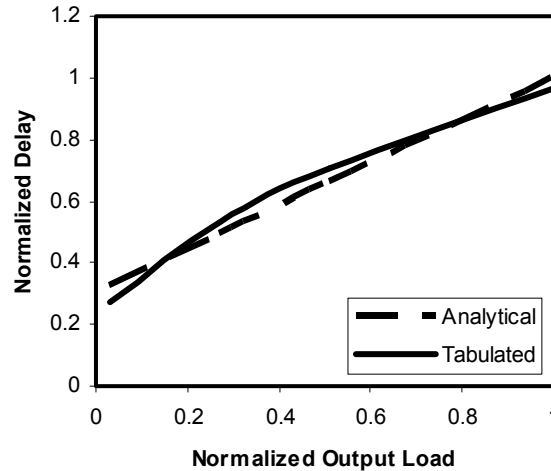


Figure 2-2. Accuracy of single-gate-fit posynomial gate delay model compared to tabulated values.

Figure 2-2 plots normalized values of an inverter delay predicted by the analytical posynomial gate delay model compared with tabulated data points. The inverter is loaded with a fixed capacitance with value corresponding to the horizontal axis of the plot. The Inverter is also driven with a fixed input slope. Coefficients for the analytical delay model were generated from data points of a single gate and have less fitting errors compared to the general case where these coefficients are fitted across several gate sizes. Gate delays increase faster at small input capacitances and taper off as the gate drives larger loads. For small output loads, the gate is not loaded and exhibits a fast output transition. The high gain of this mode of operation results in an increase in effective output load capacitance due to the Miller effect, decreasing the effective drive strength of

the gate. At high output loads, the effect of Miller capacitance is reduced due to increase in output transition time, thus alleviating the extra loading at the output and increasing the effective drive strength of the gate. As observed in [Zlatanovici06], the analytical delay curve is pessimistic for low and high output loads but optimistic for mid-range loads. The maximum fitting error observed across the output loads is 19% and occurs at low output loads.

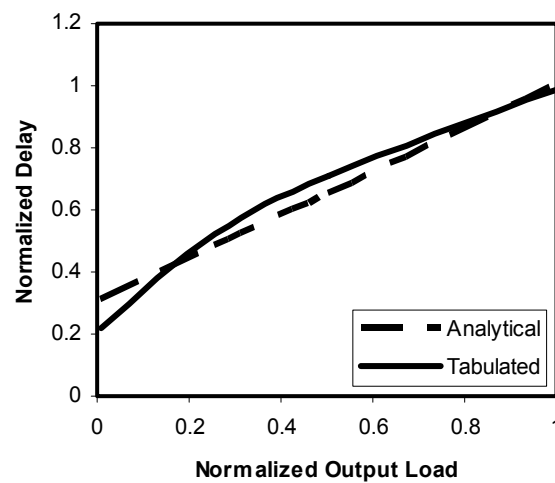


Figure 2-3. Accuracy of multiple-gate-fit posynomial gate delay model compared to tabulated values.

Figure 2-3 plots normalized values of a gate delay predicted by an analytical posynomial model fitted using tabulated delays from multiple gate sizes of the same logic function. This model represents the case corresponding to the gate sizing problem where the optimizer is required to choose an optimal gate size and needs to model gate delays across multiple gate sizes. During the least-squares fitting process, the coefficient for the constant term,  $p$  had to be pegged at a very small positive number in order to meet the constraint that all coefficients in a posynomial are positive. This limitation as well as the

least-square fit across multiple gate sizes results in a worst case fit error of 41% which occurs at low output loads.

Equations (2-4) and (2-5) assume that parameters such as  $p$  and  $g$  remain constant with change in gate size  $W_i$ . To first order, this should be the case and it is assumed that a 2X inverter will have twice the drive strength of a 1X inverter and correspondingly a 12X inverter should have twelve times the drive strength of a 1X inverter while all having the same intrinsic delay due to self-loading. Unfortunately this is not true. For example, it is necessary to use multi-finger devices when creating large gates. Although the effective transistor width of all the fingers are still similar to that of a single large transistor with equivalent drive strength, the intrinsic delay due to self-loading is reduced due to lower sidewall and bottom plate parasitic capacitance. Larger gates also have higher parasitic capacitance due to extra interconnect required to connect multiple transistors together.

[Chinnery06] employs higher order fits in an attempt to improve the accuracy of the posynomial fits and reports accuracies of 8%. He uses three monomials to model delays and slopes of the gates, with each monomial being dependent on all parameters of the gate being modeled. The increased fitting accuracy is traded off by a significant increase in the density of the matrix representation of gate delays which increases memory requirements and runtime. No matter how much effort is placed on increasing the order of posynomial fits to the data, there will still be significant modeling errors because the parameters that are being modeled are inherently non-convex. Due to the characteristic of the actual delay curve, any posynomial analytical model tends to underestimate delays across a large range of output loads, resulting in a netlist that is undersized. Based on experiments, netlists optimized using these models have a

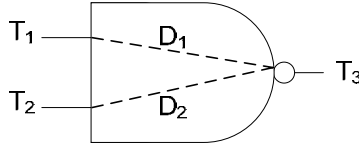
significant discrepancy in propagation delays when compared using analytical and tabulated models.

### 3.0 Combinational Circuit Optimization

This chapter provides an in-depth look at combinational circuit optimization. The optimization framework developed by [Zlatanovici06] will be reviewed and tested on a standard cell circuit. This framework, which formulates the circuit optimization problem as a geometric program, guarantees global optimality of the result and has been used to design high performance circuits such as a 250ps 64-bit Carry-Lookahead adder in a 90nm CMOS technology [Kao06]. The existing delay models that are employed in [Zlatanovici06] are accurate to a few picoseconds from tabulated values which make it useful for designing full-custom integrated circuits where the size of each transistor is a continuous variable. This framework is then extended to sizing of ASICs constructed out of standard cells. Unlike full-custom designs, standard cell designs impose more layout restrictions (such as fixed vertical pitch) that increase errors in the analytical models employed. Several improvements will be introduced to improve the quality of the results and both techniques will be evaluated on a 16-bit multiplier.

#### 3.1 Static Timer Geometric Program

The static timer geometric program optimizer framework described in [Zlatanovici06] is built around the posynomial gate delay model described in (2-4) and (2-5). A static timer formulation of delays in a combinational circuit is then used to recursively combine the arrival times at the fan-in of each gate such that the latest arrival time at each node is propagated through the combinational circuit network. For example, this recursive formulation will produce the following constraints for a 2 input NAND gate.



$$T_1 + D_1 \leq T_3$$

$$T_2 + D_2 \leq T_3$$

$$T_3 \leq D_{\text{target}}$$

Figure 3-1. Static timer delay constraints for 2-input NAND gate.

In this example,  $T_1$ ,  $T_2$ , and  $T_3$  are signal arrival times at the inputs and outputs of the gate.  $D_1$  and  $D_2$  can be expanded using (2-4) which models delays through the gate for signals arriving at each respective input. The first two constraints implicitly perform the max operation on the signals arriving through the fan-in of the gate.

By combining the gate delay posynomials using the static timer recursive formulation, the combinational circuit optimization problem can be formulated as an energy-constrained delay minimization problem as in (3-1).

$$\min_{W_i} D \text{ such that } \left\{ \begin{array}{l} E \leq E_{\max} \\ C_{in} \leq C_{in,\max} \\ W_i \geq W_{i,\min} \\ W_i \leq W_{i,\max} \\ t_{\text{slope},j} \leq t_{\text{slope},\max} \\ T_j = 0 \text{ for primary inputs} \\ T_j \leq D \text{ for all } j \\ T_j + D_i \leq T_i \text{ for } j \in FI(i) \end{array} \right.$$

(3-1)

By changing the objective function to the energy of the circuit, the optimization can be reformulated as a delay constrained energy minimization problem as in (3-2).

$$\min_{W_i} E \text{ such that } \left\{ \begin{array}{l} D \leq D_{max} \\ C_{in} \leq C_{in,max} \\ W_i \geq W_{i,min} \\ W_i \leq W_{i,max} \\ t_{slope,j} \leq t_{slope,max} \\ T_j = 0 \text{ for primary inputs} \\ T_j \leq D \text{ for all } j \\ T_j + D_i \leq T_i \text{ for } j \in FI(i) \end{array} \right. \quad (3-2)$$

$W_i$  refers to the gate sizes, which are the variables that are being optimized. We impose limits on these variables,  $W_{i,min}$  and  $W_{i,max}$ , so that the optimizer only chooses gates that are available in the standard cell library. The input capacitance,  $C_{in}$ , is limited to  $C_{in,max}$  to meet the maximum capacitance constraints of the gates driving the primary inputs.

Energy consumption in integrated circuits can be broken down into two components: dynamic and static energy. Dynamic energy is consumed to perform a particular computation. Most of dynamic energy in digital integrated circuits is consumed in charging and discharging capacitances on nodes of the circuit, which can be modeled using the following equation.

$$E_{dynamic} = \sum_{all\ nodes} \alpha_i C_i V_{dd}^2 \quad (3-3)$$

$\alpha_i$  refers to the activity factor of the node, or how many times a node switches during a computation, while  $C_i$  is the capacitance switched at the node. Another contributor to dynamic energy is the short-circuit current, which is direct conduction between  $V_{dd}$  and ground that occurs for a brief period of time during the output transition when both PMOS and NMOS networks are conducting. We have chosen to lump this

together with capacitive switching energy by increasing the capacitive switching energy by a fixed percentage.

Static or leakage energy is consumed by the circuit regardless of the computation that is being carried out. It is contributing to a significant portion of energy consumption of circuits in highly scaled technologies and should be accounted for in energy minimization. Static energy is also highly dependent on the state of the inputs of the gate. For example, in the series NMOS stack of a 2-input NAND gate, leakage is significantly reduced if the bottom transistor is in the off-state because this raises the voltage on the source of the top transistor, thus reverse-biasing it and reducing the leakage current. The state dependence of leakage can be accounted for if state probabilities of the inputs of the gates are known through simulation. Since this information might not be available during the initial phase of the design, we can get an estimate of leakage energy by assuming an equal probability for each state.

The main sources of leakage are sub-threshold conduction, gate leakage, and junction leakage. Since there are a variety of leakage mechanisms, accounting analytically for all sources of leakage in a standard cell would increase the computational complexity of the optimization problem significantly. Out of the three sources of leakage outlined above, only junction leakage is highly layout dependent. The other two leakage sources scale with gate size or gate drive strengths. We can therefore assume a linear dependence between leakage energy and gate sizing because sub-threshold conduction and gate leakage are typically much greater than junction leakage [Keshavarzi00]. However, this assumption might not hold true for future scaled technologies which exhibit increased layout-induced variations as measured by [Pang06].

$$E_{static} = \sum_{all\ gates} P_{leak,i} W_i D \quad (3-4)$$

Equation (3-4) provides the energy model for static energy of the circuit.  $P_{leak,i}$  corresponds to the leakage power of a unit-sized version of the corresponding gate. This is scaled by the size of the gate ( $W_i$ ) and delay of the circuit ( $D$ ) to obtain the energy consumed per operation.

## 3.2 Optimization Using Global Fit Parameters

This section presents the results of applying the optimization framework in [Zlatanovici06], which is described in the previous section, to a 16-bit multiplier synthesized using a commercial tool. Global fit parameters of the standard cell gates were used to model gate delays and slopes according to the posynomial model described in Section 2.2. These global parameters were produced by performing a least-squares fit based on all tabulated values available for a particular logic gate. These tabulated values span all gate sizes, output load capacitances, and input transition slopes in which the gate is characterized. Interconnect delays were annotated at every node using wireload models included in the standard cell library. This interconnect model consists of a fixed resistor driving a capacitor at each node, as illustrated in Figure 3-2.

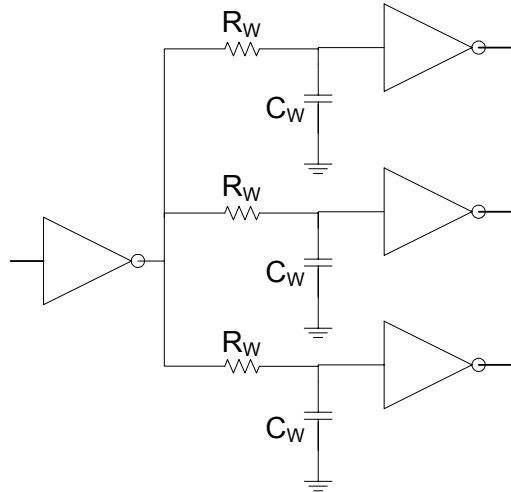


Figure 3-2. Interconnect delay model.

Table 3-1: Results of circuit optimization using global models.

Item	Delay (ns)
Synthesis tool timing constraint target	0.88
Delay of synthesized netlist estimated using global fit parameters	1.10
Unconstrained delay minimization	1.00

Table 3-1 lists the delays of the netlist before and after an unconstrained delay minimization was carried out. The purpose of such an optimization is to obtain the fastest possible circuit without any concern for energy limits. The 16-bit multiplier was first synthesized with a very tight timing constraint of 0.88 ns. The delay was then estimated using global fit parameters in order to measure the delay improvement achieved by the optimizer. These results match predictions that a continuously sized netlist is more optimal than a discretely sized netlist. By allowing continuous gate sizes, the optimizer is able to find a design point that is better than the original netlist. Indeed it is a global

optimum with respect to the global fit parameters used in the optimizer, although it comes with an uncertainty that depends on the quality of the model.

To answer the question of whether the result obtained by optimization using global fit parameters is actually better than the result of commercial synthesis, the analytical solution needs to be evaluated using accurate tabulated values. [Zlatanovici06] introduces the concept of a near-optimal boundary which is established by evaluating the netlist obtained from the analytical solution using tabulated models. It is deemed near-optimal because it is obtained from an analytical solution with errors in the model. The quality of the result, or the distance from the optimal point is thus subject to the magnitude of errors that are accrued in the analytical model. Unfortunately tabulated data might not exist for gate sizes chosen by the analytical solution because standard cell libraries only provide gates in discrete sizes. The next section describes a technique for evaluating this boundary using local fit parameters.

### 3.3 Local Fit Parameters

This section introduces the concept of local fit parameters which can be used as an approximation for a continuously sized gate. Global parameters suffer from gross fitting errors because they are a least-squares fit of a concave function to a linear equation such as (2-4). Since we are only interested in evaluating gate delays and output transition slopes of a gate at one point described by an input slope, output load, and gate size, we can reduce fitting errors by limiting the dataset to data points from tabulated values that are close to the current operating condition of the gate. For example Table 3-2 and Table 3-3 are normalized 2D lookup tables for an actual gate with two different sizes.

**Table 3-2: Gate delay lookup table for actual gate with size 1 (data normalized to delay of 2X gate with 0.0031 ns input slope and 1 unit normalized output capacitance).**

Input Slope (ns)	Normalized Output Load Capacitance			
	1	8.26	15.7	30.7
0.0031	1.33	5.05	8.84	16.5
0.286	10.3	20.0	26.2	35.6
0.596	17.0	30.1	38.9	52.2

**Table 3-3: Gate delay lookup table for actual gate with size 2 (data normalized to delay of 2X gate with 0.0031 ns input slope and 1 unit normalized output capacitance).**

Input Slope (ns)	Normalized Output Load Capacitance			
	1	15.9	31.0	61.3
0.0031	1	4.87	8.74	16.5
0.286	9.82	20.1	26.5	35.9
0.596	17.2	31.0	40.0	53.0

The shaded entries in Table 3-2 and Table 3-3 are used to generate a local fit of the parameters to obtain an estimate of the delay of a gate of size 1.36 driving a load of 20 units with an input slope of 0.1 ns. Equations (2-4) and (2-5) are augmented as follows with interaction terms to match the model used in Synopsys NLDM for a better delay estimate.

$$t_D = p + g \cdot \frac{\sum K_i \cdot W_i}{K_{current} \cdot W_{current}} + \eta \cdot t_{slope,in} + \zeta \cdot \frac{\sum K_i \cdot W_i}{K_{current} \cdot W_{current}} \cdot t_{slope,in}$$

(3-5)

$$t_{slope,out} = \lambda + \mu \cdot \frac{\sum K_i \cdot W_i}{K_{current} \cdot W_{current}} + \nu \cdot t_{slope,in} + \beta \cdot \frac{\sum K_i \cdot W_i}{K_{current} \cdot W_{current}} \cdot t_{slope,in}$$

(3-6)

These two equations are still posynomials if all the coefficients are positive. To evaluate the near-optimality boundary of the analytical solution, the convex posynomial constraint is relaxed into a polynomial constraint because these parameters are not being used for convex optimization.

**Table 3-4: Evaluation of optimization results using local fit parameters.**

Item	Delay (ns)
Unconstrained delay minimization – global fit parameters	1.00
Near-optimality boundary – local fit parameters	1.115

Evaluating the delay of the netlist using local fit parameters is the best possible estimate of actual delay short of actually designing a custom standard cell with gate sizes that match the analytical solution. Based on results listed in Table 3-1 and Table 3-4, the optimization framework using global fit parameters fails to produce a better solution. The global fit parameters upon which the optimization framework is built upon suffers from excessive fitting errors which eventually limit the utility of the tool.

### 3.4 Optimization Using Local Fit Parameters

The results of the optimization can be significantly improved by generating parameters for the gate delay and output slope models using the local fit parameters introduced in Section 3.3 and then iteratively solving the geometric program until the optimizer arrives at a satisfactory design point. New local fit parameters are generated in each iteration based on current values of the input slope, output load, and gate size of each gate in the netlist in order to provide the best estimate of delays. This algorithm is based on the technique suggested in [Li04] which limits the optimization to a local design space where non-convex models can be approximated as convex functions. Models are then fit according to this local design space and the analog circuit is then optimized. This fitting and optimization step is successively applied on narrower design spaces until the design space is small enough. While the algorithm in [Li04] was applied to analog circuits, the technique can be successfully applied on digital circuits with a few modifications. Figure 3-3 gives a general diagram of the local-fit iterative optimization algorithm.

The algorithm starts by first solving the unconstrained delay minimization geometric program using global fit parameters just as in Section 3.2. The purpose of this step is to get a good initial starting point for further optimizations using local fit parameters. More unconstrained delay minimizations are then performed using local fit parameters until the delay of the circuit meets or exceeds the target delay requirement. The unconstrained delay minimization step is performed first before energy minimization

to ensure that the optimization is feasible with respect to timing, based on evaluation using local fit parameters.

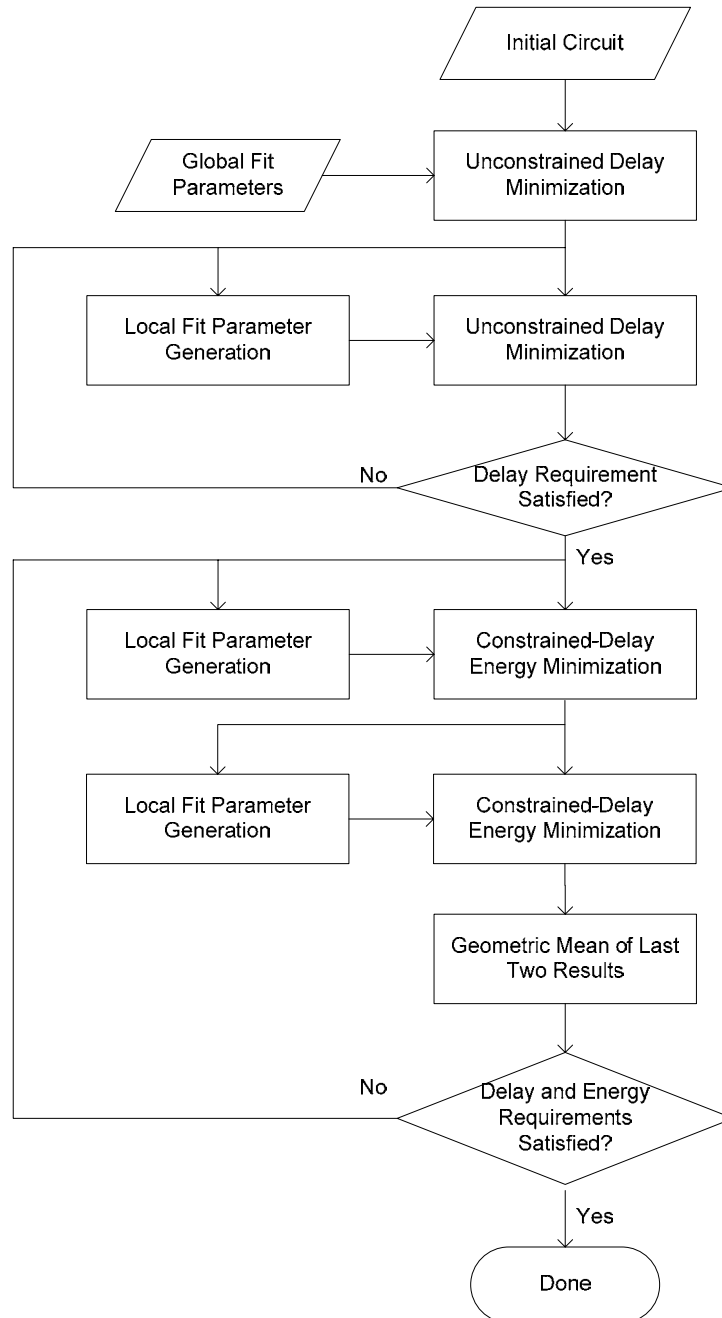


Figure 3-3. Local fit iterative optimization algorithm.

Next, two successive, delay-constrained energy minimization optimizations are performed using local fit parameters and the geometric mean of the solution of these two steps are taken. The purpose of taking the geometric mean is to overcome over-estimation and under-estimation of gate delays which is discussed in Section 3.5. The energy and delay of the circuit is then evaluated to check if the solution is good enough. If the solution is not satisfactory, the delay-constrained energy minimization step is repeated again until a satisfactory solution is achieved. Due to the localized accuracy of the local fit parameters, the optimization steps can only arrive at global optimum with respect to the local fit parameters used. This is why the energy minimization step is repeated using continuously refined local fit parameters until a satisfactory solution is obtained.

The local-fit iterative optimization algorithm was applied on a 16-bit synthesized multiplier netlist with the goal of minimizing energy at the synthesized delay target of 0.88 ns. A tighter delay requirement of 0.86 ns was passed to the optimizer to account for fitting errors in the posynomial local fit parameters of gate delay. Figure 3-4 plots energy and delay measurements of several data points during the optimization process. As illustrated by the solid lines on the plot, the algorithm exceeded the delay requirement of 0.86 ns after two iterations of delay minimization.

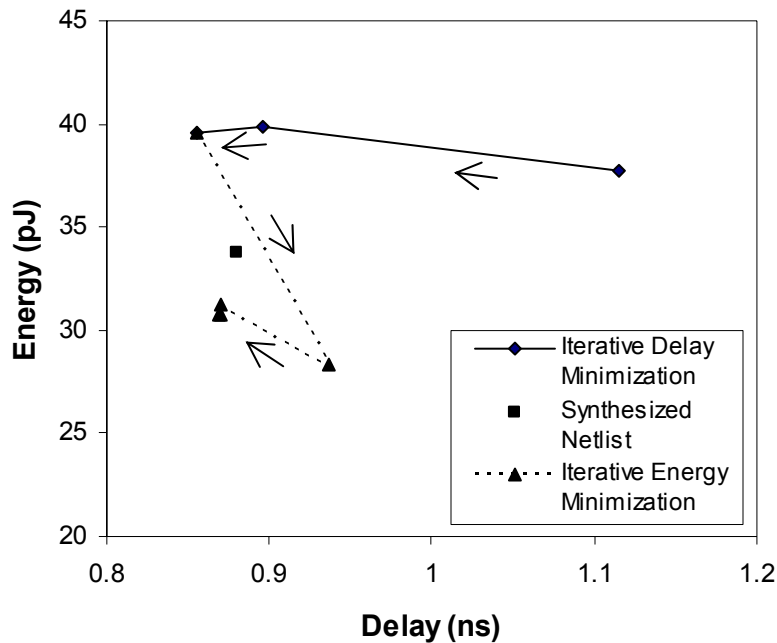


Figure 3-4. Progress of iterative local-fit optimization in energy-delay space.

Figure 3-4 also plots in dotted lines data points generated during the energy minimization loop after the geometric mean is taken. As illustrated, the algorithm arrives at a solution that is better than the commercially synthesized netlist after two iterations in the energy minimization loop.

### 3.5 Localized Convex Fit of Concave Gate Delay Models

This section investigates convergence issues resulting from applying a localized convex approximation to a concave gate delay model. Figure 3-5 illustrates the trajectory of the optimization algorithm as successive unconstrained delay minimizations are applied. As demonstrated in the graph, the optimization converges to a relatively stable design point centered around (39.7 pJ, 0.86 ns) in the energy-delay space after 2

iterations. The change in energy during each successive optimization is less than 0.5% of the total energy of the design.

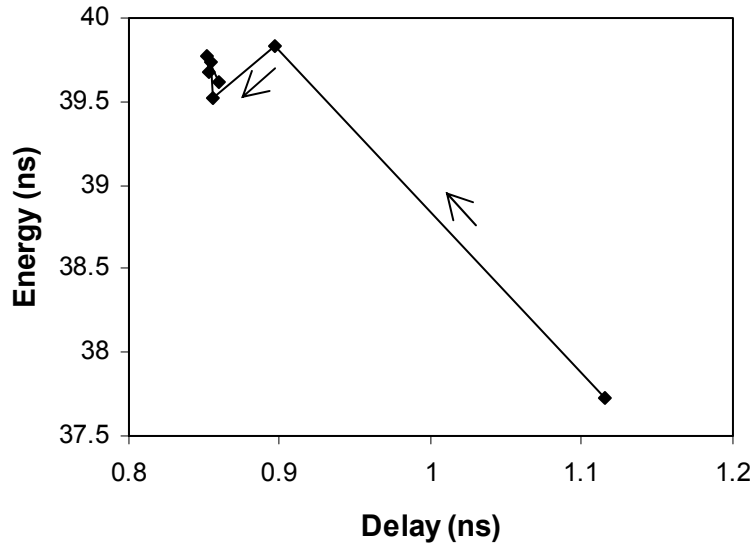


Figure 3-5. Convergence of iterative local fit delay minimization.

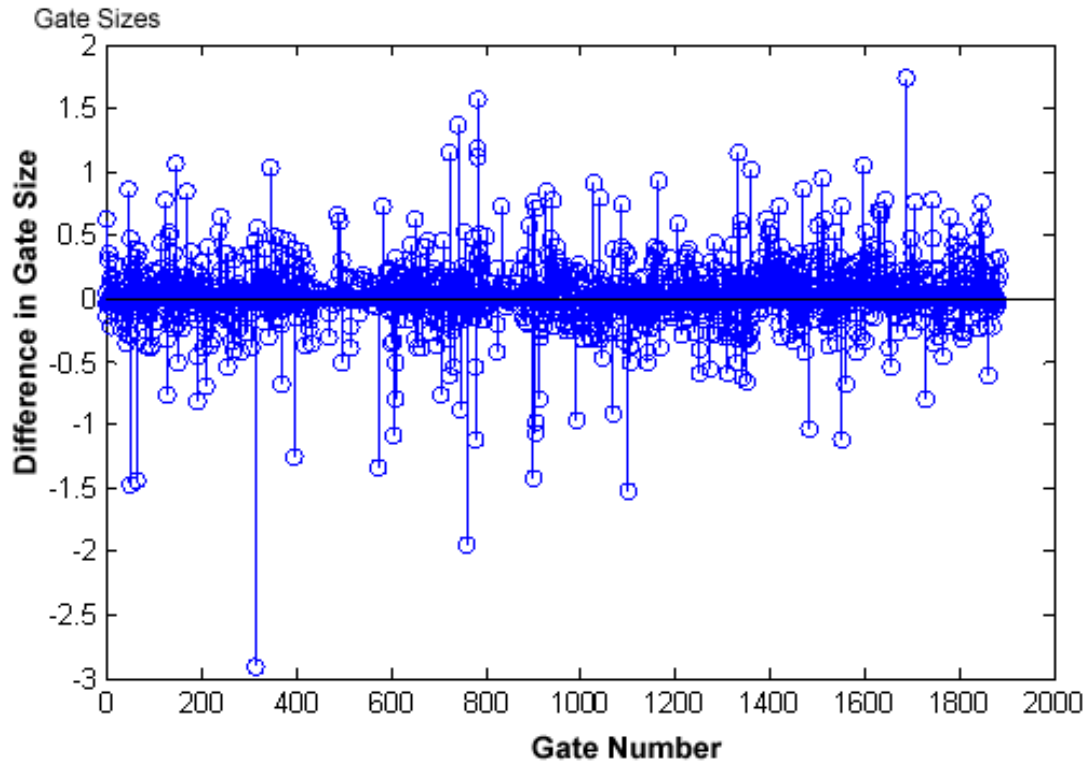


Figure 3-6. Difference in gate sizes between successive delay minimization runs.

Figure 3-6 is a stem plot of the difference in gate sizes of the last two delay minimization runs plotted in Figure 3-5. Although some gates are resized by as much as 3 gate sizes, most gates remain the same size as the average absolute value of gate size change is 0.13 gate sizes. Even though the gate sizes are not fixed, the delay and energy of the circuit is still relatively stable. This indicates that the optima for minimum delay gate sizing is wide and is in agreement with theoretical results in [Sutherland99]. The unconstrained delay minimization using local fit parameters is able to arrive at a design point that is much better than the global optimum point obtained using global fit parameters. This motivates the use of local fit parameters for gate sizing geometric programs at least for solving the unconstrained delay minimization problem.

Figure 3-7 illustrates data points that are produced as the result of successive constrained-delay energy minimization runs starting from the initial point indicated by a bold square. In each iteration, the algorithm oscillates between fast, high energy data points and slow, lower energy data points. Figure 3-8 plots the difference in gate sizes produced by the optimizer for two successive constrained-delay energy minimization runs. The optimizer sizes up all the gates during one run and sizes down the gates by approximately the same amount during the next iteration.

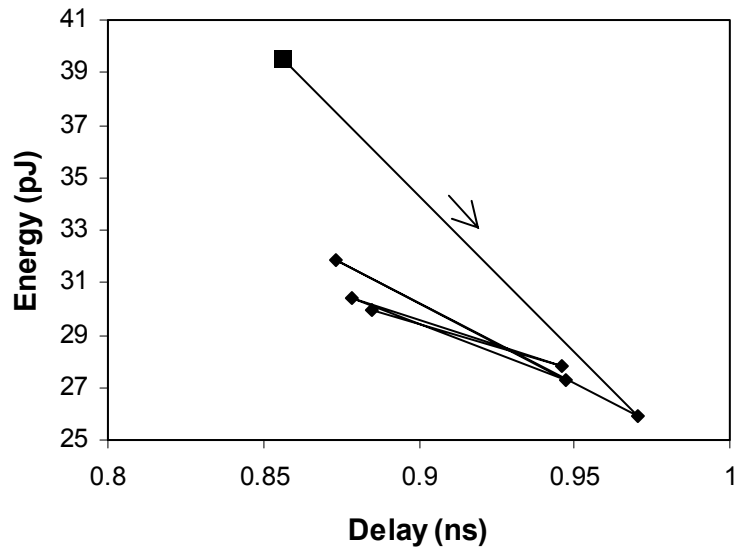


Figure 3-7. Convergence of iterative local fit constrained delay energy minimization.

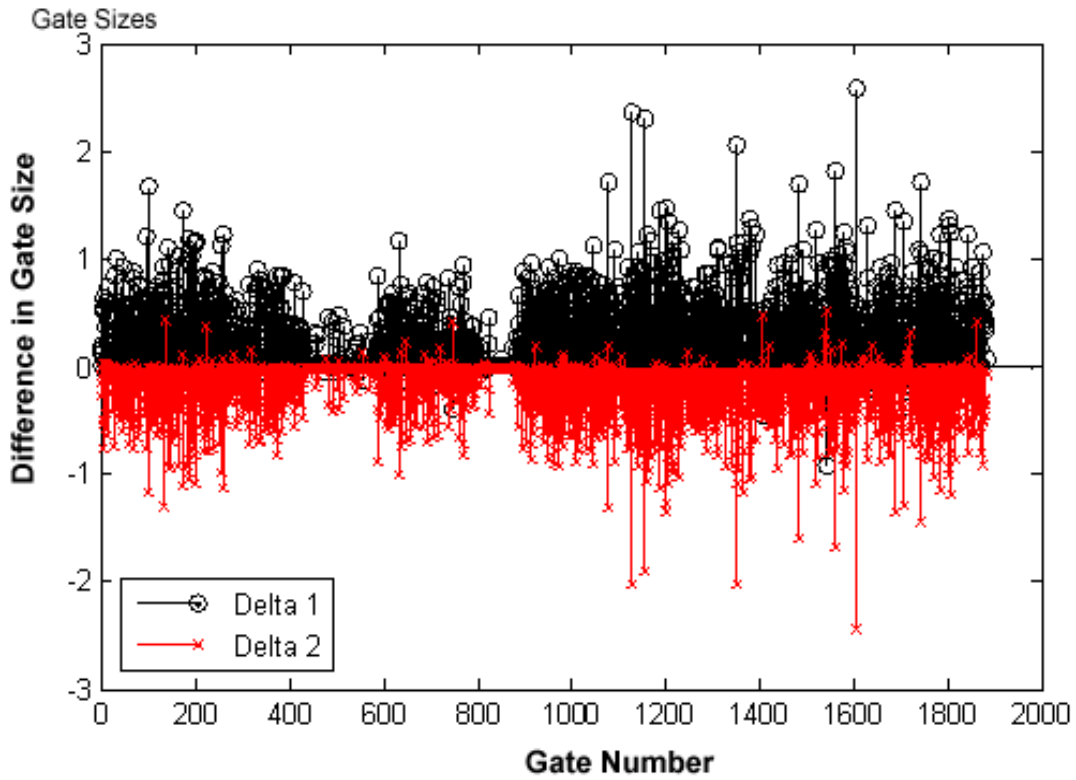


Figure 3-8. Difference in gate sizes for two successive energy minimization runs.

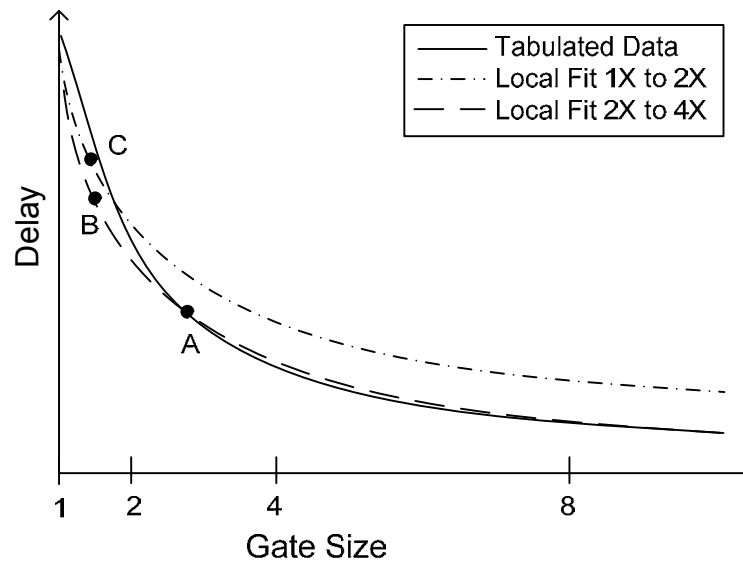


Figure 3-9. Local posynomial fitting of gate delay with gate sizing.

Figure 3-9 illustrates how the optimizer generates a local fit of the gate delays. In each iteration, the optimizer generates parameters to define a posynomial equation that best describes the gate delay around the local operating point. Unlike a piecewise function, only one curve is used to characterize the whole range of load capacitances. The assumption is that the optimizer will only explore a local space around the current operating environment and will not require accurate characterization at the further points. While this is the case in an unconstrained-energy delay minimization geometric program where the optimizer is always trying to drive for the minimum delay, this does not hold for constrained-delay energy minimization problems. In energy minimization, the optimizer tries to remove slack in the circuit for the maximal energy savings.

Starting from the initial design point with minimum delay (indicated by a solid box in Figure 3-7) the optimizer sizes down all the gates based on a trajectory set by the local fit parameters. This process can be visualized for one gate in Figure 3-9. Starting from point A, the optimizer tries to remove excess slack by downsizing the gate. The optimizer follows the trajectory of the dotted line which is described using the local fit parameters around point A. This trajectory causes the optimizer to choose a gate size such that the delay of the gate is represented by point B. However, when the next iteration of energy minimization is performed, the optimizer estimates the delay of the gate to be represented by point C instead of point B due to generation of new local fit parameters. Because the gate is now too slow, the optimizer sizes up the gate to meet delay constraints. If the gate is sized larger than 2X, the optimizer will snap to a different set of local parameters and this oscillation will continue ad infinitum. Subsequently, the

design will oscillate between a data point that's too fast and too slow, as illustrated in Figure 3-7.

The optimal gate size such that energy is minimized by removing slack is somewhere between point A and point B. The algorithm illustrated in Figure 3-3 uses a heuristic of taking the geometric mean of the last two gate sizes before checking if the optimized circuit meets delay and energy requirements. Taking the geometric mean not only gives a gate size that's between point A and point B, it also yields a gate size that is less than or equal to the midway point between A and B (arithmetic mean). Based on qualitative analysis of Figure 3-9 this is a better choice than the arithmetic mean because the absolute value of the gate delay slope decreases with increase of gate size. Selecting the arithmetic mean instead would result in an over-projection, or the choice of a gate size that is too large for the delay that is required. The harmonic mean, which is twice the product divided by the sum, provides a value that is smaller than the geometric mean but tends to be too conservative because it tracks the smaller value when differences are large.

Although the heuristic produces a netlist that is better in the energy-delay space, it does not guarantee convergence. Based on experimental data, taking the geometric mean of the last two results reduces the magnitude of gate size oscillations in subsequent iterations but oscillations are still present. [Li04] reduces the size of the search space during subsequent iterations by tightening limits of parameters to improve convergence. This will increase run time of the algorithm because the algorithm is limited in the size of the hops that it can take to arrive at the final solution. In a digital circuit optimization problem where thousands of gates are being sized concurrently, the run times will be

prohibitive. The optimization problem might even become infeasible due to limitations imposed on the search space.

In this chapter, the combinational circuit optimization framework for full-custom circuits developed in [Zlatanovici06] was extended to allow optimization of standard cell integrated circuits. Local fit parameters were introduced to overcome non-idealities that arise due to the use of fixed standard cell templates of gates and also errors in delay modeling caused by fitting errors. It was demonstrated that using global fit parameters for standard cell circuit optimization results in such gross errors that the optimized circuit is actually slower than the original circuit. These errors were reduced by employing an iterative optimization algorithm which uses local fit parameters generated from the last optimization run. The final optimized circuit was demonstrated to be superior to global fit parameter optimization both in delay and energy. We have therefore demonstrated an algorithm for obtaining a better netlist composed of gates with continuous sizes using more accurate iterative timing models. The next chapter will analyze how this continuous solution can be used to guide the rounding process and obtain a discretely sized netlist with performance and energy comparable to commercial synthesis.

## 4.0 Snapping Back Continuous Gate Sizes to Discrete Gate Sizes

The gate delay model used in the convex optimization framework assumes that continuous gate sizes are available. Standard cells are typically available in discrete sizes with more complete libraries offering smaller step sizes. As mentioned by [Boyd05], the gate sizing optimization problem with discrete sets of gate sizes is a combinatorial optimization problem which is difficult to solve. By relaxing the discrete sizing constraint and assuming continuous sizes are available, the gate sizing problem can be solved for global optimum as a GP while taking into account the various interactions of all the design parameters in the optimization problem. Hopefully this solution will yield a good starting point upon which rounding algorithms can be applied. The alternative to performing a rounding step is to implement the ASIC using automated standard cell library generators such as [Hashimoto04] which are able to generate gates of arbitrary sizes. This chapter provides a survey of rounding heuristics suggested in the literature and concludes with a comparison of the discrete solution produced using a simple greedy algorithm and the solution generated by a commercial synthesis tool.

### 4.1 Simple Rounding Algorithms

A trivial rounding algorithm involves rounding to the closest discrete gate size. [Hu07] observes that this yields poor results for a sparse standard cell library, resulting in timing violations of several nanoseconds in a 0.13  $\mu\text{m}$  technology. This observation agrees with results obtained by applying the algorithm on our test circuits. Although the

optimum for gate sizing is fairly wide, the distance to the closest gate is too far. These rounding errors compound along the critical path and end up degrading the output arrival time severely. This problem is exacerbated by the fact that gate sizes are usually available in geometric increments to create an efficient standard cell library. Due to this geometric progression of gate sizes, the distance between the continuous solution and the closest discrete size is further increased, especially for larger gate sizes. These rounding errors are more pronounced in gates with multiple fanouts where the difference in the fanout capacitive load between the continuous and discrete rounded solution might be large due to the multiplicity of load gates being rounded.

Another simple rounding algorithm involves rounding up all gates to the nearest discrete size. Although this algorithm yields delays that are faster than the round-to-closest algorithm, the switching energy of the circuit is significantly higher due to the larger capacitance being switched. This discrete sized solution is also far from the optimal minimum in energy because the path delays of the circuit are spread wider which indicates a waste of energy stemming from paths that are faster than they need to be. Based on these observations, it is obvious that simple rounding algorithms do not work well for solving the problem of going from a continuous solution to a discrete solution mainly due to the fact that standard cell libraries only offer sparse choices of gate sizes.

## 4.2 Round and Resize Algorithm

[Boyd07] proposes a more sophisticated rounding algorithm which solves the geometric program multiple times but rounds and fixes parameters to discrete sizes if they are within 10% of the closest discrete gate size. The assumption of this algorithm is

that this rounding process does little harm to the quality of the solution due to the proximity of the continuous gate size to the discrete size. Any reduction or increase in slack introduced by this selective snapping process will be absorbed by the rest of the gates which will be resized during the next iteration. Based on our experiments, this algorithm often gets stuck in an infeasible state after a few iterations due to the fixed gates violating slope constraints. At this point, either the slope constraints can be relaxed or an algorithm can be used to fix the violated constraints to make the geometric program feasible before proceeding. The algorithm usually terminates leaving behind a significant number of gates that are still continuously sized because they are not driven close enough to a discrete size to be rounded and fixed. The remaining gates can then be rounded using one of the simple rounding algorithms mentioned before.

Compared to the trivial rounding algorithms, the round and resize algorithm produces a final netlist that is faster than just rounding up and consumes less energy. Unfortunately this algorithm suffers from the same issues as the previous algorithms and is unable to satisfy timing constraints due to the final rounding step that is used to bring remaining continuous sizes to discrete values. This algorithm also inherits the inaccuracies of the posynomial gate delay models that are used for the geometric program which makes it harder to achieve timing closure. The computational cost of this method is also significantly higher because the interior point solver needs to be called at every iteration to solve the geometric program.

### 4.3 Continuous Solution Guided Dynamic Programming

[Hu07] proposes the technique of searching for the optimal solution systematically using dynamic programming but limiting the search space by guiding the search using the continuous solution. A dynamic programming approach to gate sizing can obtain the optimal solution for the discrete gate sizing problem but is computationally expensive due to the need to investigate every gate size at each node. The algorithm in [Hu07] takes advantage of the continuous solution to limit the choices of gate sizes to those with drive strengths close to the continuous gate size. The algorithm also tries to minimize both delay and area by using these two metrics for pruning solutions at each node. This timing driven pruning process helps ensure that the algorithm produces a solution that meets timing constraints while the continuous solution guided selection of gates ensures that the energy consumption of the circuit is close to optimal.

[Hu07] compares this algorithm with the sizing algorithm proposed in [Coudert96] that uses a multi-dimensional descent method and demonstrates reduction in area by 9% to 31% while meeting timing constraints. However, this algorithm requires up to twice the runtime of the approach in [Coudert96]. This extra runtime is in addition to the time required to obtain the continuous solution. The ability of this algorithm to produce a better solution without a drastic increase in runtime demonstrates the benefit of using a continuous gate solution for guiding discrete gate sizing. It also provides experimental evidence that the optimal solution utilizing discrete gates is usually close to the continuously sized solution.

## 4.4 Continuous Solution Guided Greedy Sizing

While the dynamic programming approach to gate rounding appears to be the best solution to obtain an optimal discretely sized solution from a continuous solution, our intention is not to repeat work that has already been done by [Hu07]. We would like to evaluate in a simple way whether the continuous solution obtained using the algorithm covered in Section 3 provides a good starting point upon which advanced rounding algorithms such as [Hu07] can be applied to obtain a good solution. A simple greedy sizing algorithm similar to [Lin90] was implemented to round the continuous solution and results were compared with solutions obtained from a commercial tool.

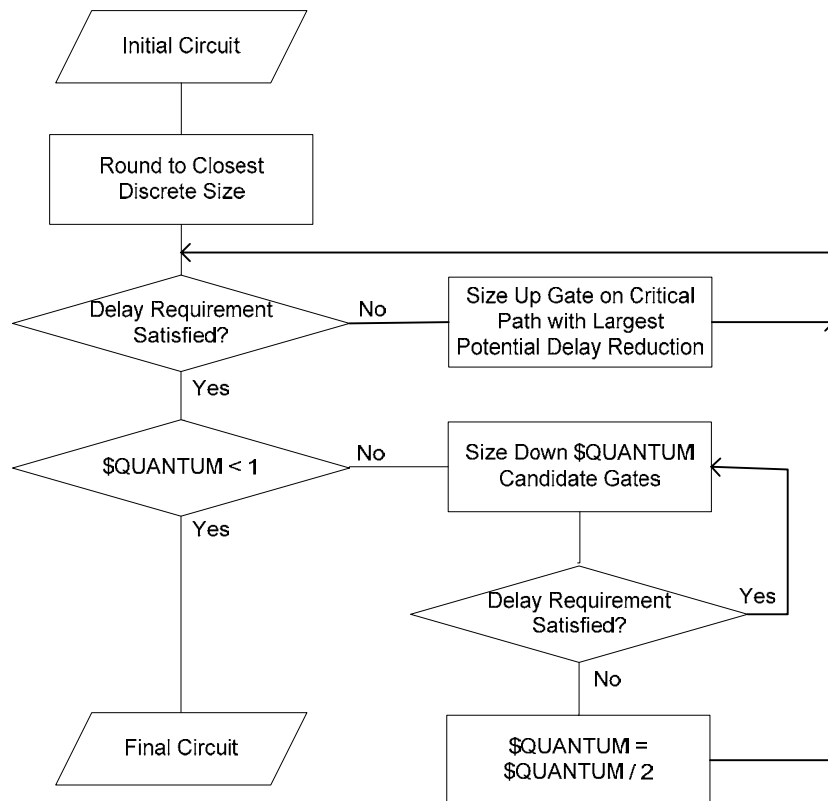


Figure 4-1. Greedy sizing algorithm.

Figure 4-1 outlines the simple greedy sizing algorithm. The algorithm first takes the continuous solution and rounds the gates to the closest discrete size available. The algorithm then alternates between two phases: meet delay constraints and minimize energy. In the optimization loop for meeting delay constraints, the algorithm traverses the critical path and selects one gate to be upsized that will result in maximal reduction of delay or increase in critical path slack. Changing the size of a gate will change the input arrival times and slopes at the resized gate and also slopes and delays of all paths in the fan-out cone of the resized gate. Moreover, a single sizing can change the sensitivity of the paths, therefore affecting the slack of all the gates [Coudert96]. Calculating the exact change in slack from upsizing a gate could potentially require a static timing analysis of the whole circuit. To minimize this computational complexity, the greedy sizing algorithm only recalculates the arrival times and slopes of the fan-in nets. This is illustrated in Figure 4-2. The change in slack of the gate is then the difference between  $t_{arrv3}'$  and  $t_{arrv3}$ . This is similar to the approach in [Coudert96] where the delay is only evaluated within a sub-network around the gate being resized. This delay minimization step is then repeated until delay requirements are met or no candidate gates were found, in which case the algorithm exits due to an infeasible timing constraint.

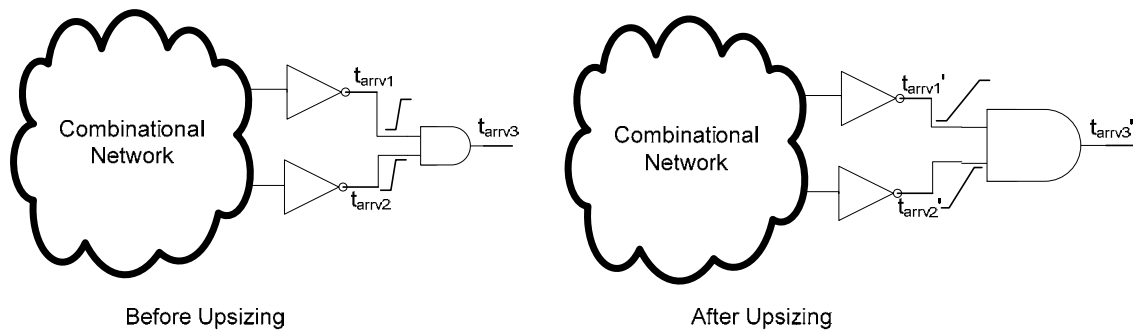


Figure 4-2. Evaluation of change in slack due to upsizing of a gate.

In the energy minimization loop, the algorithm sizes down QUANTUM number of gates from the netlist that have the largest slack. The choice of QUANTUM affects the convergence of the algorithm. A large value of QUANTUM will likely cause severe degradation in delays, requiring constant alternation between minimization of delay and minimization of energy loops while a small value of QUANTUM results in slow convergence of the algorithm. Based on suggestions in [Lin90] a practical value of 32 was used in our experiments. Whenever the timing requirement is violated within the energy minimization loop, the algorithm halves the value of QUANTUM so that future energy minimization runs will be less greedy.

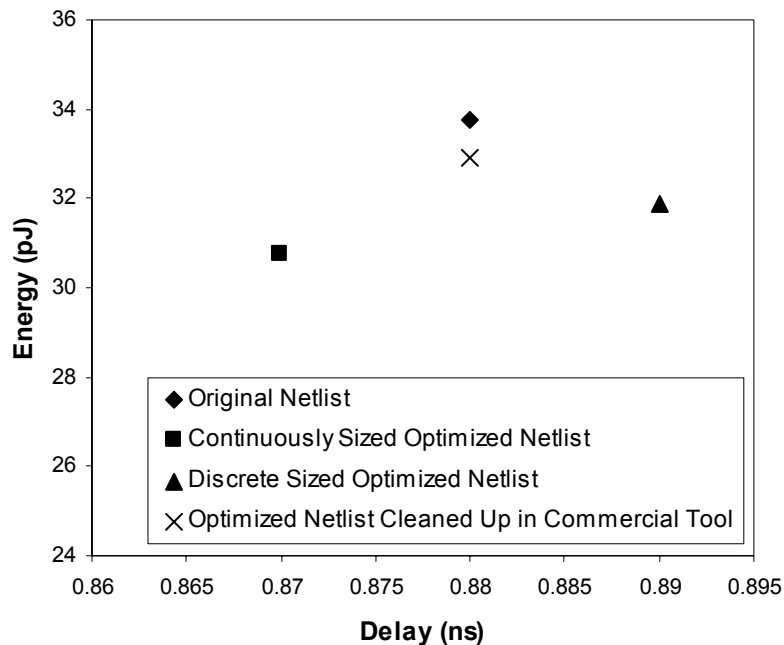


Figure 4-3. Results of rounding using greedy sizing algorithm.

Figure 4-3 plots the results that were obtained using the continuous solution guided greedy sizing algorithm in the energy-delay space. Starting from the original

netlist indicated with the diamond, the optimizer was able to obtain a continuous solution, indicated by the square box, which is better in both energy and delay. The greedy sizing algorithm was then applied on the continuous netlist to obtain a solution indicated by the triangle. The algorithm was only able to arrive at a solution with a delay of 0.89ns even though 0.88 ns was required. The greedy sizing algorithm terminated within the delay minimization loop because it was unable to find a candidate for upsizing. The energy of the solution produced by the greedy sizing algorithm was 6% lower than the original netlist with a negligible 10 ps timing violation. At this point, the discrete sized netlist was input into the same commercial synthesis tool that produced the original netlist and “cleaned-up” to meet the 0.88 ns timing requirement. By taking the optimized netlist as a starting point, the commercial tool was able to arrive at a design with 2% lower energy and still meet timing requirements.

Although the greedy sizing algorithm was unable to meet timing constraints, it was still valuable in providing a starting point for further optimization using a commercial synthesis tool. An analysis of the gate sizes returned by the greedy sizing algorithm indicates that more than 99% of the gates in the discrete solution are within one discrete size from the continuous solution. This supports the motivation for using a dynamic programming method guided by the continuous solution for producing a discrete solution because the discrete solution is usually close to the continuous solution. The algorithm of [Hu07] will produce a better discrete solution than this greedy sizing method because the dynamic programming approach searches a wider space around the continuous solution. However, the result demonstrated in this section using a simple

greedy sizing algorithm is sufficient to demonstrate the quality of the continuous solution generated using our iterative optimization algorithm.

## 5.0 Optimizing Sequential Circuits

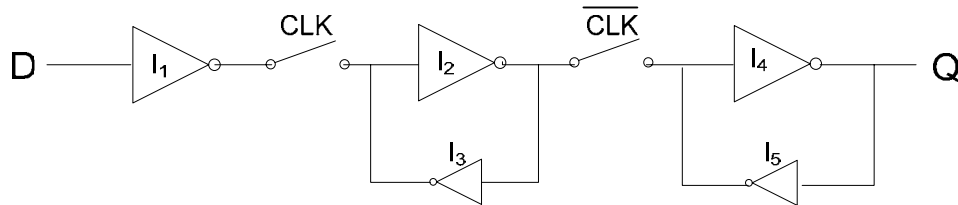
A sequential circuit is a network of logic gates and storage elements which communicates with its environment through primary inputs and primary outputs. The storage elements are clocked by a signal which defines when new data is stored in the element. Pipelining is an effective way to increase the throughput of a circuit by inserting sequential elements at cut-sets along the circuit graph. High performance circuits that are performance limited due to power dissipation constraints are usually pipelined and could benefit most from an optimization framework that supports sequential elements. Therefore a robust power-performance optimization tool should be able to account for the setup and hold requirements of these sequential elements to enable joint optimization of these sequential elements with other parameters.

This chapter first analyzes the timing properties of flip-flops, followed by a review of the existing framework for supporting sequential elements developed by [Zlatanovici06] including suggestions on how to improve the modeling of flip-flop delay. Finally, a technique for efficiently solving geometric programs of large sequential circuits is covered.

### 5.1 Flip-flop Gate Model

Sequential elements such as flip-flops are significantly more complex compared to combinational logic gates. Three of the most important timing characteristics of a flip-flop are its setup, hold, and clock-output delays which are measured with respect to a clock edge. The setup time is the period that the data input (D) must be valid before the clock transition, while the hold time is the period where the data input must remain valid

after the clock edge [Rabaey03]. The clock-output delay is the worst case delay after the arrival of the clock edge before the signal at the output (Q) is ready. Since most flip-flops have negative hold times for most practical values of input slope on CLK, hold times will not be considered in our gate model.



**Figure 5-1. Schematic of a D flip-flop.**

Figure 5-1 illustrates the schematic of a flip-flop constructed using master-slave registers. Most commercial flip-flops are usually more complicated than this, but this simple schematic is sufficient for our analysis. Inverter I<sub>1</sub> is used to buffer the flip-flop from the gate driving the D input for isolation. As a rule of thumb, this inverter is usually not upsized for flip-flops with larger drive strengths since it only functions as a buffer. This was verified to hold true for flip-flops available in multiple commercial standard cell libraries.

Even if this is not the case, the loss in optimality from having a circuit that is not sized for the correct load is minimal. This is evidenced in Figure 6-9, which plots energy-delay tradeoff curves for a 161-bit leading zero detector. The proximity of the energy-delay tradeoff curves even as load capacitance is changed by an order of magnitude indicates a weak sensitivity of energy and delay to load capacitance. The circuit blocks can thus be resized on a second pass once flip-flop sizes have been allocated without too much loss in optimality. Based on this analysis, we can safely assume that flip-flops present the same capacitive load to the driving circuit regardless of their drive strengths.

[Zlatanovici06] makes the assumption that all flip-flops have a ratio between output load to input load ( $C_L/C_{in}$ ) of 1. This results in an extremely pessimistic model of the input capacitance and causes energy wasteful upsizing of gates on the input cone of the flip-flop to drive the overestimated capacitance.

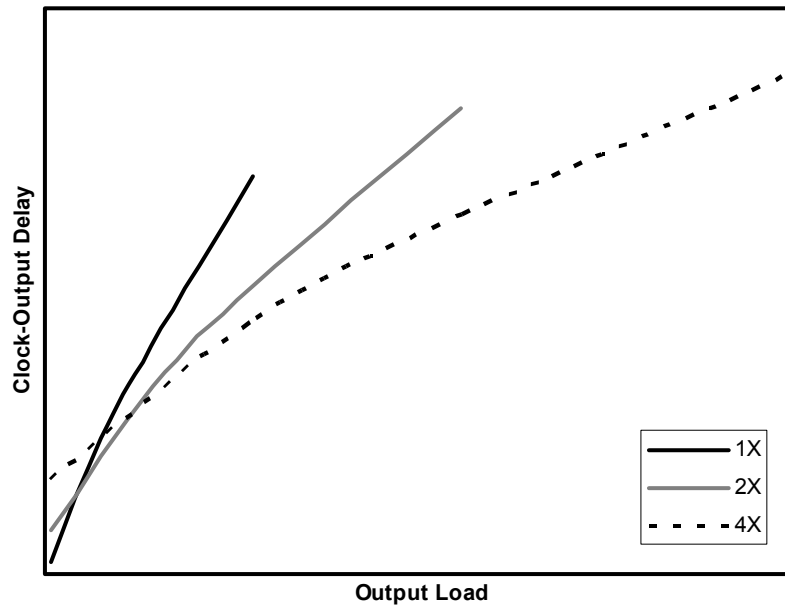


Figure 5-2. Clock-output delay of flip-flops with different drive strengths.

Figure 5-2 plots the clock-output delay at different output loads for flip-flops of various drive strengths. Just like the propagation delays of logic gates, the clock-output delays of flip-flops are weakly concave functions of output load and could therefore make best use of the local fit parameter extraction technique introduced in Section 3.3 to improve the accuracy of delay calculations. It is also observed that the intrinsic delay (clock-output delay when no output load is present) of the gates differ significantly between the gates giving rise to even more errors in delay estimation if the gate delays were modeled using global fit parameters. Furthermore, the slopes at the output of the

flip-flops are significantly worse than the output slope of a combinational logic gate with equivalent drive strength.

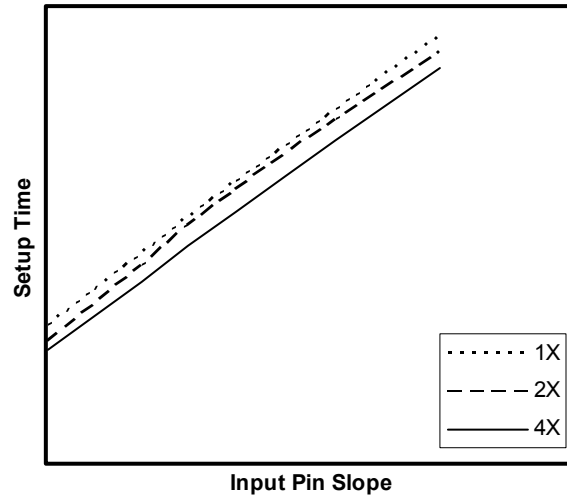


Figure 5-3. Setup time of flip-flops with different drive strengths.

Figure 5-3 plots the dependence of the setup time of flip-flops of various drive strengths on input pin (D) slopes. This plot was produced under the assumption that signal slopes on the CLK port are fixed. Based on this plot, it can be concluded that setup times for each flip-flop can be approximated very well using a linear function such as (5-1).

$$t_{setup} = \lambda + v \cdot t_{slope,in} \tag{5-1}$$

Since the difference between the setup times of the flip-flops is at most 10%, this parameter can be reasonably estimated across all drive strengths using two parameters as in (5-1), with no dependence on gate size, without too much sacrifice in accuracy. This simplification, together with the observation that flip-flops present the same capacitive

load to the driving circuit regardless of drive strength provides a convenient means of decoupling the sizing of a flip-flop from the delay of the circuit driving it. The delay of the combinational circuit driving flip-flops can thus be calculated a priori without knowledge of the sizes of the flip-flops.

For our analysis, we have adopted a simple linear model of flip-flop dynamic and static energy that scales linearly with flip-flop sizing. Clock distribution is not included in our analysis but could be potentially modeled as a posynomial that is dependent on total area of the circuit, H-tree partitioning, and a clock driver ratio, such as done by [Liu94]. Such information should be available to the optimizer since the optimizer is executed post-synthesis. Finally, only a single variety of flip-flop with three drive strengths is considered in our model. Different varieties can be considered at an architectural level using the heuristic described in Section 6.7 for combining multiple circuit blocks.

## 5.2 Optimization of Circuits with Sequential Elements

Optimization problems similar to those covered in Section 3 can be formulated by simply replacing the “delay” of the combinational circuit with the “cycle time” of the sequential circuit. A sequential circuit can be represented as a directed acyclic graph (DAG)  $G(V,E)$  as introduced by [Leiserson91]. Each element  $v_i$  in  $V$  corresponds to a combinational logic gate in the sequential circuit with two special source and sink nodes representing inputs and outputs of the circuit respectively. Each edge  $e_i$  in  $E$  corresponds to an interconnection between two vertices in  $V$ . To represent the presence of sequential elements in the DAG, edges are annotated with weights which indicate the number of

registers that a signal needs to propagate through when traversing through a particular edge.

Regardless of how many sequential elements are present on each edge, a signal propagating through the combinational circuit network needs to arrive early enough at the source vertex of the edge such that the sum of the delay with setup time of the sequential element included is less than the required cycle time. Furthermore a signal is launched into the destination vertex of the edge at the beginning of the clock period, after a delay corresponding to the clock-output delay of the sequential element. The static timer GP formulation introduced in Section 3.1 can thus be amended to support sequential elements by considering sequential elements as special source and sink nodes embedded within the graph with extra timing requirements associated with setup times and propagation delays.

The sequential circuit optimizer developed by [Zlatanovici06] assumes that all inputs and outputs have sequential elements. While this simplifies the GP formulation by making the timing requirements of the circuit explicitly dependent only on the cycle time of the circuit, this is usually not the case in practical pipelined circuits. Most pipelined circuits are optimized as a block defined around the boundary with input delays and output delays. Input delay is defined as a fixed propagation delay after the arrival of the clock before the input signal is available to the combinational circuit, and output delay is defined as the duration of time after the signal is launched at the output pipeline stages of the circuit before the signal is required at the outputs. Input delays can be included in the GP formulation of sequential circuits by augmenting delays of vertices with fan-in edges coming from primary inputs with the corresponding input delays. Output delays can be

accounted for by adding extra constraint functions corresponding to each output delay requirement. Equation (5-2) lists the objective and constraint functions for the minimum-energy sequential circuit resizing GP accounting for both input and output delays.

$$\min_{W_i} E \text{ such that } \left\{ \begin{array}{l} D \leq T_{cycle} \\ C_{in} \leq C_{in,max} \\ W_i \geq W_{i,min} \\ W_i \leq W_{i,max} \\ t_{slope,j} \leq t_{slope,max} \\ T_j = T_{input\ delay,j} \text{ for primary inputs} \\ T_j \leq D \text{ for all } j \text{ gates driving flip – flops} \\ T_j \leq T_{output\ delay,j} \text{ for all } j \text{ gates driving primary outputs} \\ T_j + D_i \leq T_i \text{ for } j \in FI(i) \end{array} \right. \quad (5-2)$$

The optimizer of [Zlatanovici06] assigns a fixed propagation delay and setup time to every sequential element in the circuit. While this approximation is appropriate for slower sequential circuits, this simplification can result in poorly designed circuits, especially in high performance pipelined circuits where the setup time and propagation delays of the sequential elements account for a significant portion of the cycle time. This could either result in unachievable target cycle times due to pessimistic estimates of these fixed delays or wasteful upsizing in the fan-out of sequential elements due to an over-estimation of delays. These modeling errors can be reduced by improving on the timing model of the flip-flops, based on the results observed in Section 5.1.

The setup time of all flip-flops in the library can be reasonably estimated using a simple linear model such as (5-1). However, unlike the setup times of flip-flops, propagation delays of flip-flops are highly dependent on sizing and are difficult to fit to a general model, as observed in Section 5.1. To avoid having to use a more complicated model to capture the effects of sizing on propagation delay, the optimizer was augmented

with an extra step to traverse all flip-flops and iteratively cycle through the choices of flip-flops from the library, looking for the cell which minimizes delay. Local fit parameters corresponding to the selected sizes are then used in the GP. The area overhead from selecting an over-designed flip-flop is minimal because flip-flops are more complicated cells and have significantly larger footprints. The increase in area from increasing drive strengths is small compared to the footprint of the whole cell. Furthermore flip-flops present approximately similar load and setup time to the fan-in regardless of sizing, which decouples sizing of the fan-in from sizing of the flip-flop.

### 5.3 Efficiently Optimizing Large Sequential Circuits

The main motivation for GP modeling of digital integrated circuits is the great efficiency with which optimization problems of this special form can be solved [Boyd07]. Furthermore the static timing formulation of node delays is sparse and only depends on a few variables [Zlatanovici06]. This property can be exploited by the interior point optimizer to speed up computation of the solution. While this optimization technique shows great promise, it is worthwhile to investigate how the complexity of the computation scales with the size of the problem.

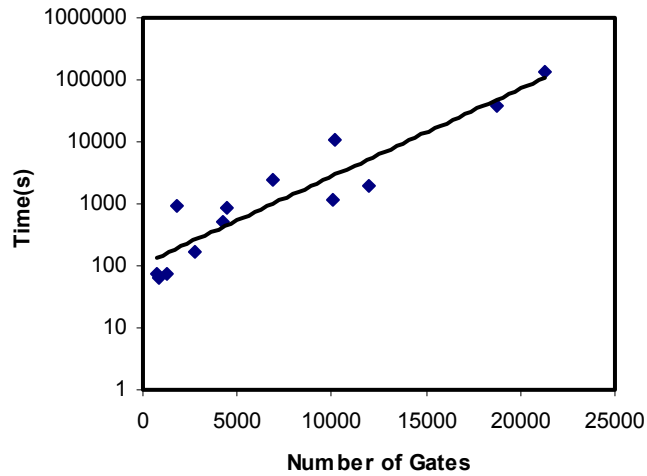


Figure 5-4. Computation time versus number of gates in the digital circuit.

Figure 5-4 plots the time required to perform an unbounded energy, delay minimization for a few test circuits with different gate counts. These experiments were conducted on a 2.6 GHz dual Opteron workstation with 2 GB of memory using only one processor. Note that each of these circuits used for runtime measurements have different circuit topologies with different circuit complexities. It is therefore possible for two circuits with similar gate counts to have different runtimes. Based on observation of Figure 5-4, the runtime of the interior point optimizer increases exponentially with gate count. The computation time of the optimizer is reasonable for datapath circuit blocks such as adders and multipliers which are usually only a few thousand gates. Unfortunately, the exponential increase in runtime makes it prohibitive to optimize complete datapaths.

To speedup the optimization of large sequential circuits, these circuits can be divided into smaller sub-circuits with boundaries defined by the cut-set of sequential elements. According to Section 5.1, flip-flops with buffered inputs present similar

loading and timing requirements to the fan-in regardless of the size of the gate. Even if this assumption is not valid, the loss in optimality is not significant. This means that sizing of the fan-in network and fan-out network of the flip-flop is decoupled and can be done separately without too much impact on the optimality of the result.

**Table 5-1: Computation time for optimizing a 53-bit multiplier.**

Netlist	Computation Time (seconds)
3 Stage Pipeline	4737
Non-pipelined	37064

Table 5-1 lists the computation time required to perform an unconstrained delay minimization of a 53-bit multiplier for two cases: 3 stage pipeline and a non-pipelined design. It is obvious that the interior point optimizer is able to take advantage of the insertion of pipeline registers which decouples interaction between pipeline stages to speed up computation.

Furthermore, we can take advantage of the independence of each pipeline stage to form islands of combinational logic driven by primary inputs and flip-flops. These sub-circuits can then be optimized concurrently on different compute resources and stitched together to produce the final solution. Ideally this should reduce the computation time by a factor of  $1/N$ , where  $N$  is the number of pipeline stages, assuming that the circuit is divided equally between all the stages. Unfortunately this is usually not the case as high performance arithmetic circuits such as carry-lookahead adders and parallel multipliers often utilize tree-type circuit topologies which have heavier logic at the inputs of the

blocks. The computation time is therefore bounded by the largest sub-circuit or the largest pipeline stage.

This speedup technique was applied on an 8 stage pipelined double-precision floating point unit and the results are summarized in Table 5-2. By dividing the full circuit into smaller sub-circuits, the computation time was reduced to one fourth the time required for the full circuit. The first two stages of the circuit consist of a 53-bit Booth encoded multiplier which reduces 27 partial products into 2 terms. Due to the wide fan-in of this multiplier architecture, a larger proportion of the full circuit is situated in the first few stages. This explains why there is a significantly larger gate count in the first two stages which directly translates to longer computation time. Although in practice, the computation speedup is not proportional to the number of pipeline stages, the reduction is still substantial and helps increase the attractiveness of performing digital circuit optimization by solving a geometric program.

**Table 5-2: Runtime required for optimization of complete double-precision FPU Circuit and separate pipeline stages.**

Netlist	Gate Count	Computation Time (s)
Full	36804	7325
Stage 1	11974	1890
Stage 2	10077	1134
Stage 3	4197	519
Stage 4	4483	856
Stage 5	2750	172
Stage 6	852	62
Stage 7	1762	920
Stage 8	709	76

## **6.0 Case Study : Optimizing Building Blocks of a Double-Precision Floating Point Unit**

This chapter presents the results of applying the standard cell optimizer developed in this work to various building blocks in a double-precision floating point unit (FPU). An FPU presents a good candidate for a case study because it is constructed out of various building blocks such as adders, multipliers, and logic units, each with different circuit architectures and logic depths. The FPU will be decomposed into these fundamental blocks to analyze the strengths and weaknesses of the optimizer when faced with a variety of circuit designs. Due to the non-optimality of the trivial rounding algorithm implemented in our work, gate sizes from the optimizer will be left as continuous sizes.

### **6.1 Double-Precision Floating Point Unit**

The FPU that will be the subject of our study takes in three IEEE double-precision floating point numbers [IEEE85], A, B, and C, and outputs  $(A \times B) + C$  in normalized double-precision format. This architecture is very similar to the FPU implemented in IBM's Power6 microprocessor [Curran06]. Figure 6-1 illustrates the block-level diagram of the FPU datapath. Only the logic blocks involved in processing the mantissa are included in this diagram. There is an exponent datapath (not illustrated) which runs in parallel with the mantissa datapath and interfaces through the input/output arrows on the left side of Figure 6-1. The building blocks which will be analyzed in this chapter are the 53-bit multiplier, 108-bit adder, 55-bit incrementer, and the full exponent datapath.

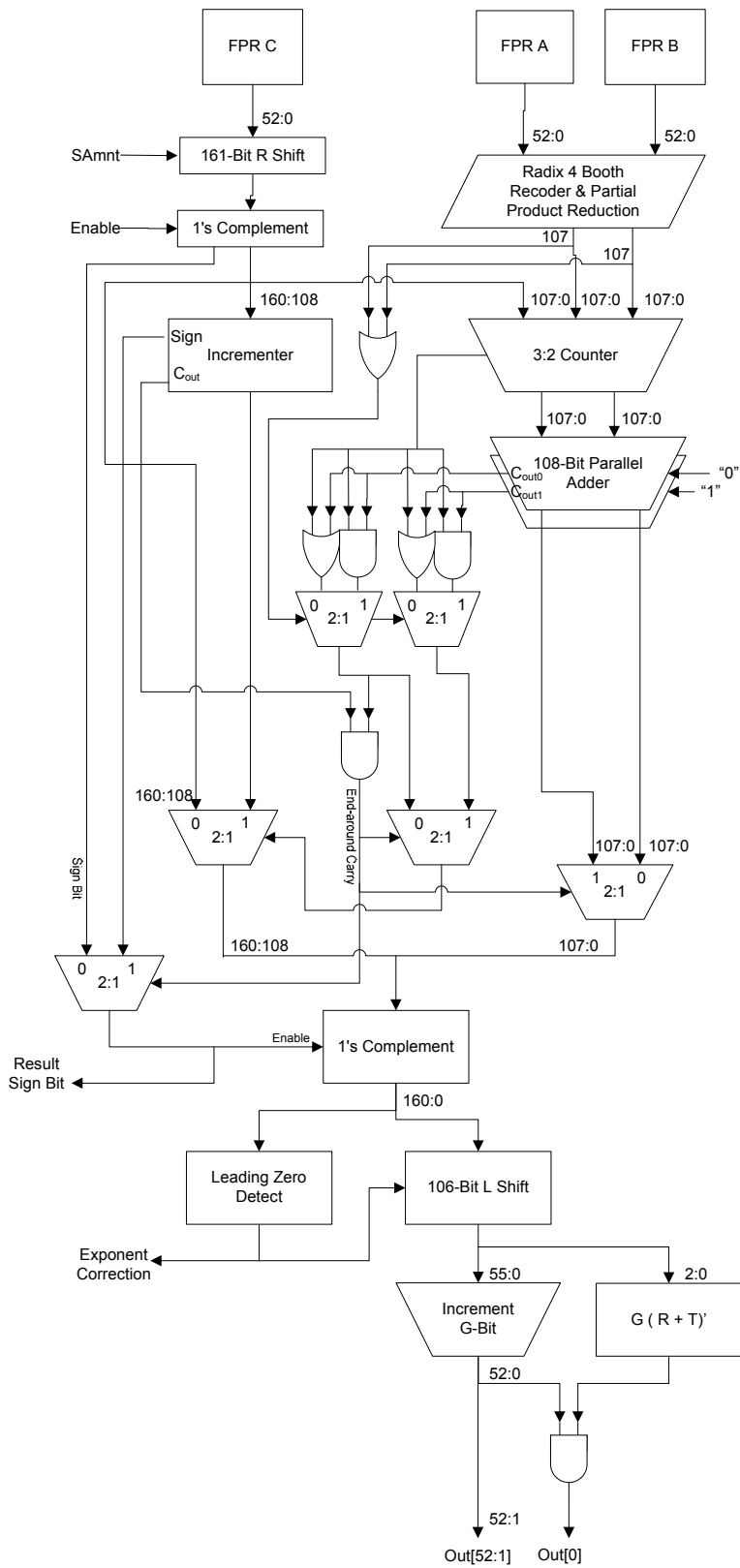


Figure 6-1. Block diagram of the mantissa datapath in a double-precision FPU.

## 6.2 Optimal Selection of Process Technology

We start our case study by finding the best process option for our design. Modern CMOS processes typically provide the circuit designer with a variety of options such as low power (LP) and general purpose (G) transistors which are optimized for different performance requirements [Tavel06]. Circuit designers also have the option of choosing from a variety of threshold voltages. An LP transistor is ideal for low performance circuits because leakage currents of these transistors are much lower than G transistors. However, LP transistors are not as fast as G transistors. Demanding high performance out of LP transistors thus ends up requiring upsizing of gates which significantly increases dynamic energy consumption. G transistors on the other hand are much faster (up to 2X), but have a lot more leakage (up to 100X). Even though G transistors have higher leakage energy, their faster speed means that smaller transistors can be used to meet performance requirements, thus resulting in lower total power.

To motivate the need to choose the right process option for the design, we conduct an experiment on a 16-bit synthesized multiplier optimally sized at different performance constraints with two different transistors: standard  $V_t$  (SVT) and high  $V_t$  (HVT). The optimization was carried out by assuming an activity factor of 1, meaning that the circuit executes a computation every cycle. The scope of this study is also limited to the case where the circuit is only implemented using one kind of  $V_t$  option. Allowing both  $V_t$  options concurrently in the optimization would convert the problem into a mixed-integer geometric program (MIGP) which is extremely difficult to solve for global optimum within a reasonable runtime without employing heuristics [Boyd07]. The results of this study are plotted in Figure 6-2.

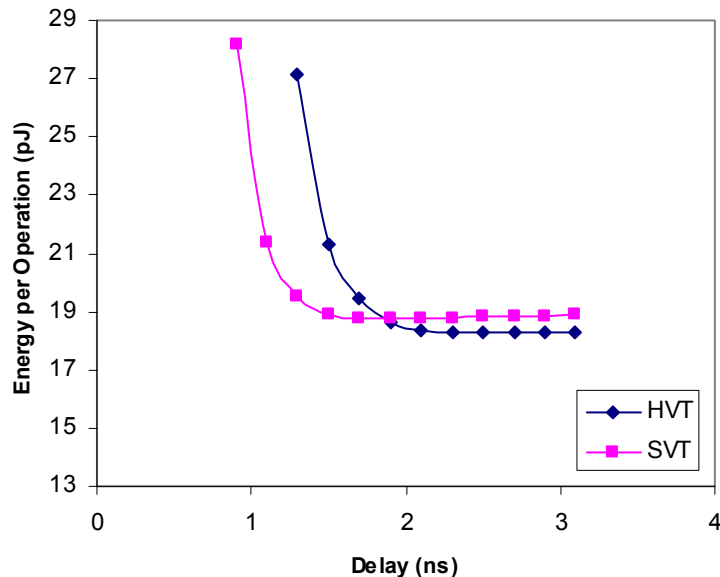


Figure 6-2. Energy-delay tradeoff curves for a 16-bit multiplier at different threshold voltages.

As illustrated in the plot, the optimally sized multiplier using SVT transistors is able to meet timing requirements from 1.0ns to 1.9ns with much lower energy compared to the HVT design. It is even able to meet performance constraints down to 0.9ns with a slight increase in energy compared to the fastest HVT design. The performance wall of the HVT design is at 1.3ns which is evident by observation of the slope of the energy-delay tradeoff curve. The optimizer up-sizes gates aggressively in order to meet performance requirements, resulting in increased dynamic energy. At the other end of the spectrum where performance requirements are lax, the SVT design ends up consuming more energy. An analysis of the netlist of both designs at 3ns indicates that all transistors are minimum sized. Since both designs are minimum sized, both designs therefore have similar dynamic energy dissipation. The SVT design ends up consuming more energy due to increased leakage energy. The results of this analysis would change drastically as

activity factor of the block is reduced, with low activity factor designs favoring higher threshold transistors to reduce leakage because the circuit sits dormant most of the time. The conclusion is that energy-performance optimization also involves proper selection of process technology. The right process option should be selected based on performance requirements. Since our FPU design is targeted towards high performance designs, we choose the SVT transistor over the HVT transistor. Note that it might also be beneficial to use HVT transistors for off-critical blocks such as the incrementer in the addend path in Figure 6-1.

### 6.3 53-Bit Multiplier

The 53-bit multiplier is the largest arithmetic block in the FPU which functions to produce the product of the mantissas of operands A and B. A straightforward implementation of the multiplier might be in the form of an array multiplier where 53 partial products are summed together. These partial products are generated by multiplying (logical AND operation) the multiplicand with each successive bit of the multiplier. The summation operation is usually carried out efficiently using a tree structure constructed out of  $(m, n)$  compressors which reduce  $m$  terms to  $n$  terms. We have chosen to reduce the partial products to two terms so that the product can be efficiently summed with the shifted addend using 3:2 counters which are also called full adders.

A 53-bit multiplier synthesized with a delay target of 1.5ns with high effort for minimizing area requires almost 20,000 standard cell gates. The tree structure of the partial product reduction circuit will also generate a dense matrix due to the relationship

between timing paths spanning the tree, requiring increased memory capacity and computation time. Based on our experiment, merely performing an unconstrained delay minimization on a non-pipelined 53-bit multiplier required 10 hours of computation time on a 2.6 GHz dual Opteron workstation using only one CPU. Fortunately, most high performance FPGAs such as [Curran06] utilize 3-stage pipelined multipliers in order to get high performance and throughput. Based on this trend, we inserted two levels of pipeline registers to divide the multiplier into 3 stages before performing optimizations. The resulting pipelined netlist required only 1.3 hours to perform an unconstrained-energy, delay minimization run, which amounts to a 10X reduction in runtime.

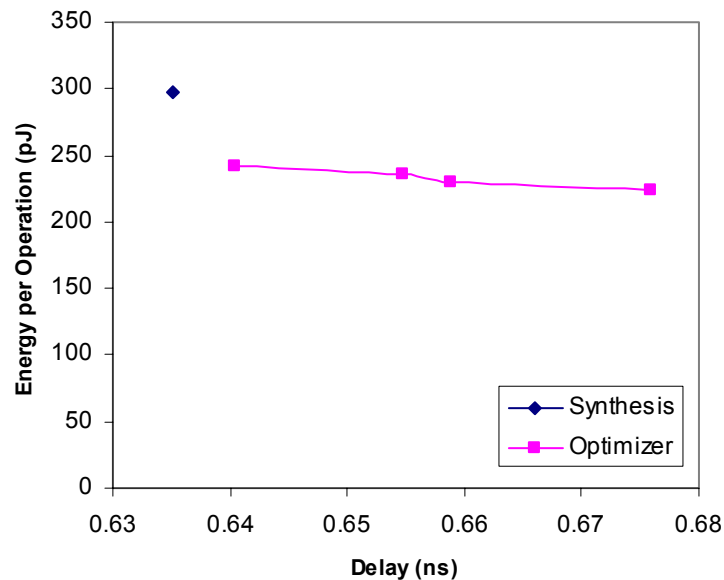


Figure 6-3. Optimization results for 53-bit multiplier.

This pipelined netlist was then used to generate data-points in Figure 6-3. Each data point required approximately 7 hours of computation time on a 2.6 GHz dual Opteron workstation, utilizing both CPUs. In comparison, the runtime for the commercial synthesis tool is 15 minutes. As illustrated in the plot, the optimizer is able to produce a

netlist that consumes 18% lower energy at 5ps slower than the synthesized netlist. The 5ps degradation in timing is negligible, compared to the magnitude of the path delay.

## 6.4 108-Bit Adder

All numbers in the mantissa path are represented in one's complement for ease in performing negation of the numbers. In order to sum two numbers in one's complement, the carry-out from the adder needs to be sent back into the carry-in of the adder. This is commonly called end-around-carry. To prevent two costly carry propagations down the carry tree of an extremely wide adder, we chose to break the feedback loop by performing two additions in parallel with different values of carry-in, as illustrated in Figure 6-1. Since both adders are exact replicas of each other, we will only consider the optimization of a single 108-bit adder. This adder was synthesized using a commercial tool with a delay target of 0.4ns and with high effort for minimizing area.

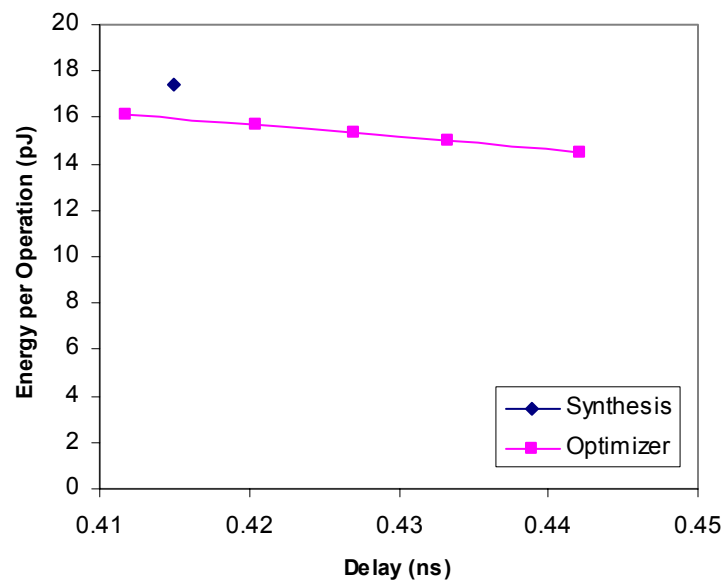
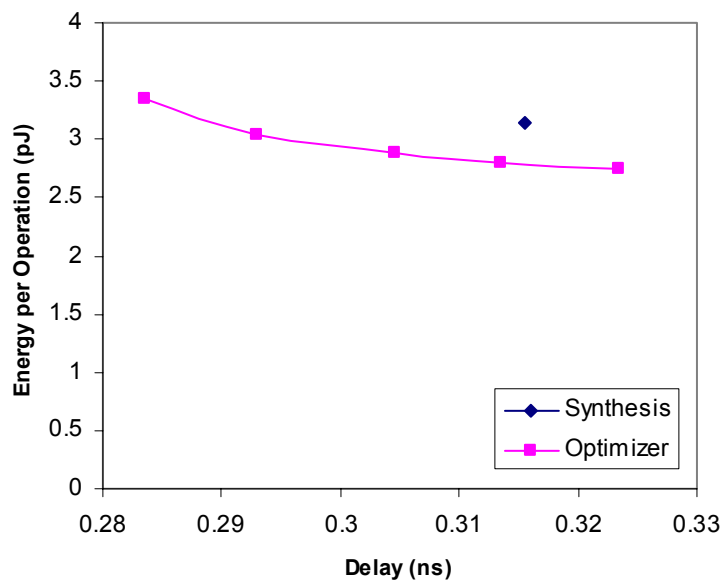


Figure 6-4. Optimization results for 108-bit adder

According to results plotted in Figure 6-4, the optimizer is able to produce a netlist that is slightly faster than the synthesized netlist with 6% lower energy. The reduction in delay and energy is only slight and can be attributed entirely to relaxation of discrete gate size constraints between the commercial tool and the optimizer. An analysis of the critical path reveals that the adder has a logic depth of 10 gates. This relatively shallow logic depth accentuates the timing errors in the analytical gate delay models and degrades the quality of the solution. Nevertheless, the quality of the solution obtained is still on par with results from commercial synthesis and sizing. The runtime to produce each point in the energy-delay space of Figure 6-4 is approximately 900 seconds on a 2.6 GHz dual Opteron workstation with 2 GB of memory utilizing both processors.

## 6.5 55-Bit Incrementer

The incrementer is used to opportunistically round the final mantissa to the next closest floating point number based on a logic operation depending on the rounding bit (R), the guard bit (G), and the sticky bit (T). The rounding logic evaluation executes in parallel with the incrementer and selects either the incremented or non-incremented result. Since it is in the critical path of the mantissa datapath, it should be optimized for high performance. The incrementer was synthesized with a tight delay constraint of 0.3ns with high effort for minimizing area.



**Figure 6-5. Optimization Results for 55-Bit Incrementer**

According to results illustrated in Figure 6-5, the optimizer is able to produce a netlist with 13% lower energy consumption at similar performance. Although part of this energy reduction may come from relaxation of discrete constraints, we are confident that this margin is large enough that the final discrete netlist obtained through rounding will still have lower energy than the original netlist. The optimizer was able to produce a much better netlist in this case, even though the 55-bit incrementer has shallower logic depth compared to the 108-bit adder, because the commercial tool might have been stuck at a local minimum due to the sequential optimization algorithm used in the tool. The runtime to produce each point in the energy-delay space of Figure 6-5 is approximately 200 seconds on a 2.6 GHz dual Opteron workstation with 2 GB of memory utilizing both processors.

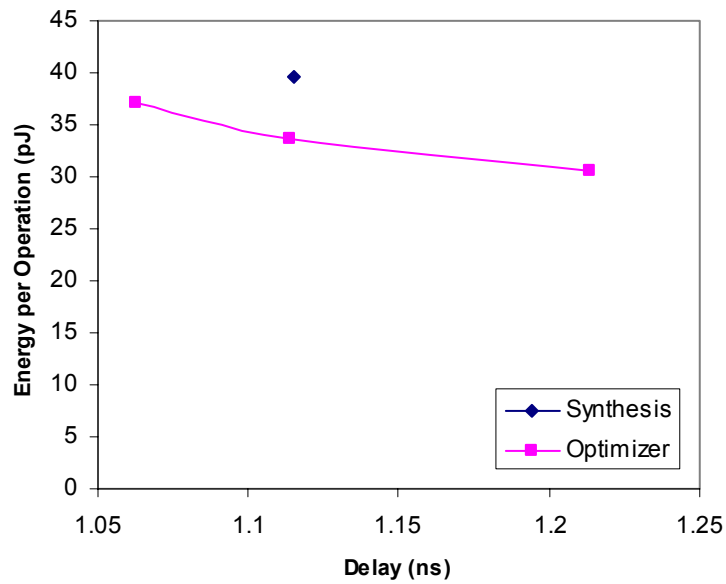
## 6.6 Exponent Datapath

The exponent datapath provides a case study to analyze performance of the optimizer when handling complex synthesized circuits. Figure 6-6 provides a Verilog description of the exponent datapath. The main purpose of the exponent datapath is to calculate the resulting exponent from taking the product of A and B. This is then compared with the exponent of C to generate signals for the mantissa datapath to align the product and addend. Based on interpretation of the Verilog description, synthesis of the exponent datapath would require a variety of adders, comparators, and multiplexors. The block was synthesized with a delay target of 1.1ns with high effort for minimizing area resulting in a synthesized netlist of 2800 logic gates.

```
module explogic(expa, expb, expc, signa, signb, signc, res_exp,
               res_sign, addend_sign);
    input [10:0] expa;
    input [10:0] expb;
    input [10:0] expc;
    input signa, signb, signc;
    wire [12:0] samnt;
    wire [12:0] prod_exp;
    output [10:0] res_exp;
    wire [7:0] actual_samnt;
    output res_sign;
    output addend_sign;

    assign prod_exp = expa + expb - (1024-1) + 53 + 3;
    assign samnt = prod_exp - expc;
    assign actual_samnt = ((samnt[12]) ? 0 : (samnt < (53*3+2) ?
        samnt : 53*3+2));
    assign res_exp = actual_samnt ? prod_exp : expc;
    assign res_sign = signa ^ signb;
    assign addend_sign = signc ^ res_sign;
endmodule
```

**Figure 6-6. Verilog description of exponent datapath.**



**Figure 6-7. Optimization results for exponent datapath.**

As illustrated in Figure 6-7, the optimizer is able to produce an optimized netlist with 17% lower energy at similar delay. Energy savings achieved by the optimizer in this case is better than other test cases in this chapter because the datapath has greater logic depth. The quality of results obtained for this test case indicates that the optimizer is capable of handling complex synthesized netlists. The runtime to produce each point in the energy-delay space of Figure 6-7 is approximately 3700 seconds on a 2.6 GHz dual Opteron workstation with 2 GB of memory utilizing both processors.

## 6.7 Combining Multiple Blocks

In the previous sections, we have applied the synthesized circuit optimizer on several building blocks and demonstrated the robustness of the optimizer in handling a variety of circuits. In this section, we will investigate methods for composing these building blocks into a complete system. The most straightforward approach would be to combine these blocks together by inserting pipeline registers between the blocks. These

blocks can then be optimized separately to drive the equivalent input capacitance of a pipeline register. Unfortunately this technique would only be feasible if the various blocks are complicated enough to be placed between two pipeline registers. In addition, the size of these blocks might be too large to be optimized within a reasonable runtime. Furthermore, when constructing complicated architectures, it is often beneficial to combine the energy-delay tradeoff curves of multiple blocks to arrive at a composite curve so that multiple architectural options can be investigated rapidly. It would therefore be worthwhile to develop a technique for combining multiple blocks that are optimized separately into a complete system.

Just like standard cells, the interface between two circuit blocks can be defined by propagation delay, input capacitance, and input slopes. Ignoring variability, an optimally sized circuit should have completely eliminated timing slack in the circuit, resulting in all output signals arriving concurrently. We can therefore describe each block with a single propagation delay without loss of optimality. Input capacitance, on the other hand, affects timing characteristics of the driving block, while input slopes affects timing of the block being driven. Putting blocks together with incompatible capacitance and slopes at the interfaces would result in less optimal designs. Figure 6-8 plots energy-delay tradeoff curves for a 161-bit leading zero detector (LZD) optimized with different maximum input capacitance ( $C_{in,max}$ ) constraints. According to this plot, the choice of  $C_{in,max}$  greatly influences energy consumption of the block, as well as the maximum performance that can be attained. The energy-delay tradeoff curve shifts down and left as  $C_{in,max}$  is increased, meaning that designs with higher  $C_{in,max}$  consume lower energy at the same

performance and are able to deliver higher performance. However, this is only one side of the story as this increased input capacitance needs to be driven by the preceding block.

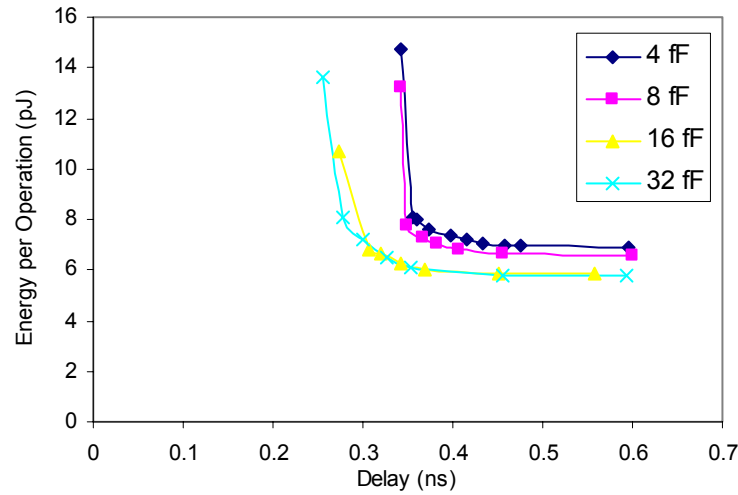


Figure 6-8. Energy-delay tradeoff curves for 161-bit leading zero detector with different  $C_{in,max}$  constraints.

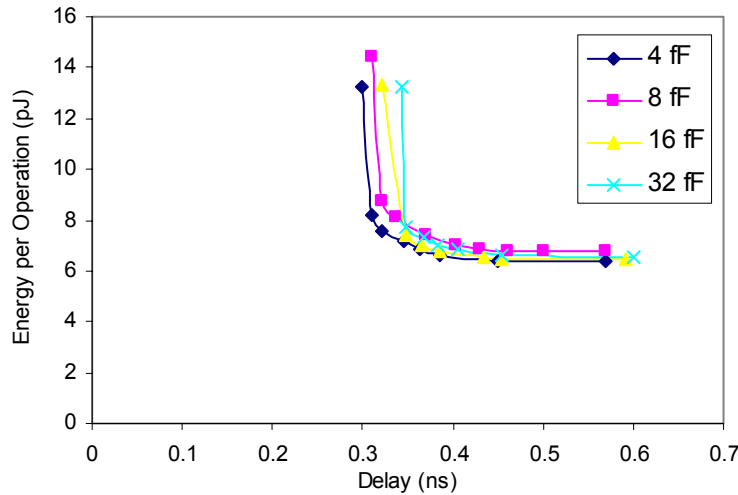
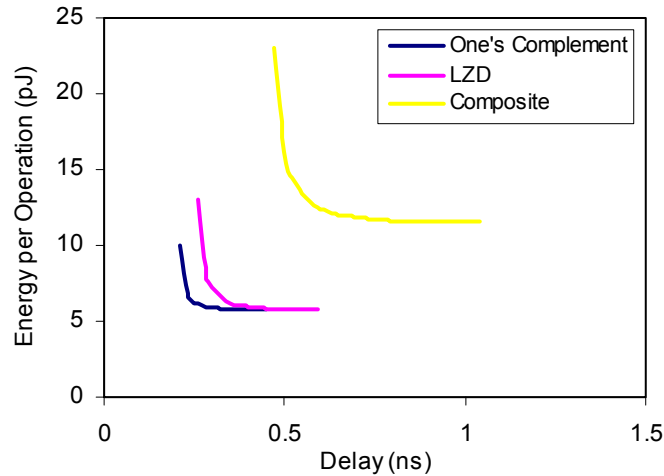


Figure 6-9. Energy-delay tradeoff curves for 161-bit leading zero detector with different  $C_{load}$  constraints.

Figure 6-9 plots energy-delay tradeoff curves with fixed  $C_{in,max}$  optimized for different output capacitance ( $C_{load}$ ). The energy-delay curves are almost similar for slower designs with delays greater than 0.35 ns. The slight difference between the curves is due to the fact that the different netlists were synthesized to drive different loads and

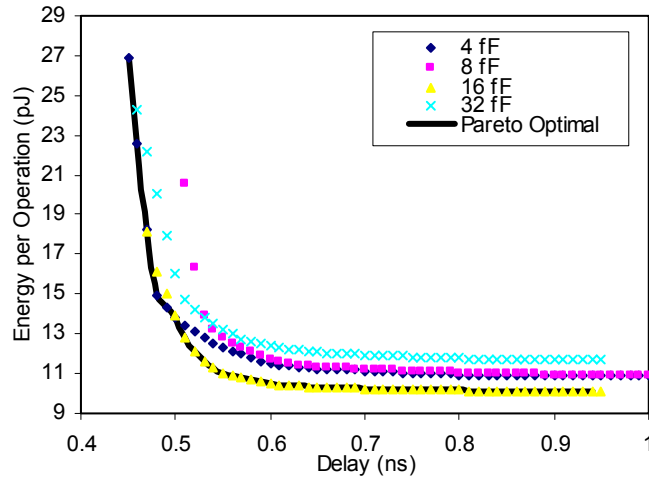
therefore have different complexities. For small delays, a lightly loaded netlist consumes lower energy and is also able to meet shorter delay requirements. Based on analysis of these two plots, it can be concluded that energy-performance optimization of multiple blocks connected together involves a balance of energy and delay between the blocks. [Markovic04] uses the notion of hardware sensitivity to formalize the tradeoff between these blocks. Hardware sensitivity is defined as the absolute gradient of energy to delay. At an optimal design-point, the hardware sensitivity of all blocks should be equal, such that an equal amount of change in energy is obtainable from each block for an equal change in delay. Such a design-point is also referred to as the Pareto optimal point.

[Markovic04] explains how the energy-delay tradeoff curves of two sequentially connected blocks can be combined into a single curve. This procedure is repeated here to demonstrate how two sequential blocks of the double-precision FPU illustrated in Figure 6-1 can be combined to form a composite energy-delay curve. The one's complement block was optimized to drive a 32 fF load while the leading zero detector was optimized with a 32 fF maximum input capacitance constraint. This ensures that the two blocks are compatible with one another. Results of this experiment are plotted in Figure 6-10.



**Figure 6-10. Energy-delay composition of two blocks connected in series.**

[Markovic04] employs the simplifying assumption of fixing  $C_{in,max}$  of each block to reduce the search space. Although this assumption might be valid for the simple example examined by the author, this does not hold true for more general circuits, as evidenced in Figure 6-8. Ideally, each block should have energy-delay curves for all combinations of  $C_{in,max}$ ,  $C_{load}$ , and input slope such that we would be able to pick an instance of each block that equalizes the hardware sensitivities. Unfortunately, since there are infinitely many combinations of these continuous variables, tabulating all these values would require an infinite amount of space. The runtime for generating these data points within the optimization loop would also be prohibitive. We have chosen to characterize each block over a fixed set of 4  $C_{in,max}$  and  $C_{load}$  capacitances resulting in 16 combinations of  $C_{in,max}$  and  $C_{load}$ . This heuristic was inspired from the fact that we are trying to solve a similar problem as standard cell mapping and sizing, albeit at a larger scale. We then perform an exhaustive search of all possible ways of interfacing the blocks together and generate a Pareto optimal curve that is the minimum of all energy-delay tradeoff curves.



**Figure 6-11. Composite energy-delay curves of all possible combinations of one's complement block followed by LZD block.**

Figure 6-11 plots the resulting composite energy-delay tradeoff curves of this heuristic. Each series of data-points is identified by their respective  $C_{in,max}$  constraint which specifies the interface between the one's complement block and the LZD block. An approximation of the Pareto optimal curve is derived from these data-points by taking the minimum of all the curves. The Pareto optimal is thus a combination between the 4 fF curve and the 16 fF curve.

A composite energy-delay curve for the entire architecture can be generated by employing a dynamic programming approach. Since the inputs of combinational logic blocks have a pre-defined  $C_{in,max}$  determined by the driver of the inputs, the algorithm would start at the inputs and generate Pareto optimal curves at the interfaces of the blocks as it propagates to the output. As the algorithm propagates through the circuit in a depth-first order, each node will be defined by a set of Pareto optimal curves corresponding to the  $C_{load}$  capacitances at which the current node is characterized for. The extent of the exhaustive search employed in this heuristic is thus limited to the set of possible

combinations between the  $C_{in,max}$  capacitances of the gates being driven at each node, which makes the algorithm computationally tractable. For each combination of  $C_{in,max}$ , the algorithm would find the closest  $C_{load}$  at which the current node is characterized at and use this Pareto optimal curve to propagate further. Such a procedure was applied on the normalization and rounding logic of a double-precision FPU illustrated in Figure 6-12.

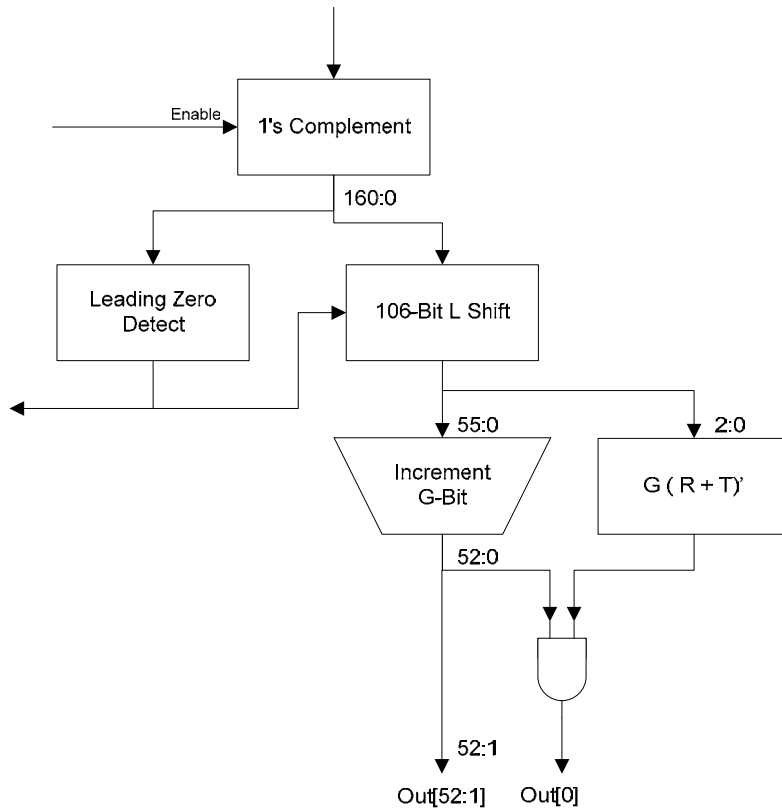
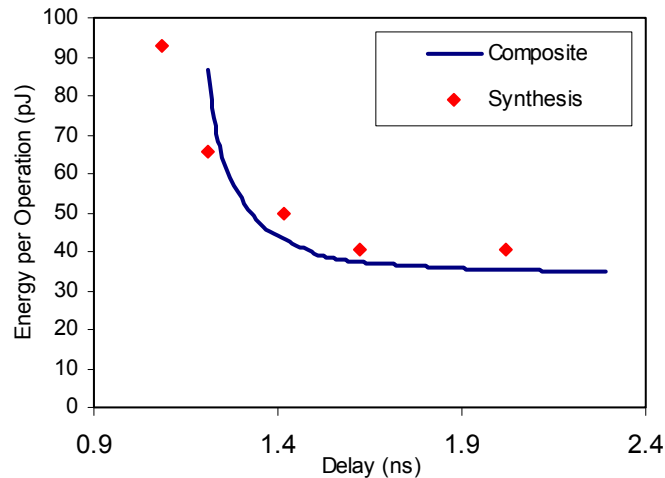


Figure 6-12. Normalization and rounding logic of a double-precision FPU.



**Figure 6-13. Comparison between composite curve and synthesis design-points of FPU normalization-rounding logic.**

Figure 6-13 plots the composite energy-delay tradeoff curve generated from individual energy-delay curves of the blocks. The normalization and rounding logic was also synthesized using a commercial synthesis tool in order to compare the quality of results. These data-points are plotted as individual points on Figure 6-13. The heuristic is able to generate a composite curve that follows the synthesis data-points very well. At longer delay constraints, the solution obtained from the composite energy-delay curve is superior to that obtained from best-effort synthesis by up to 11%. This reduction in energy consumption at the same performance can be attributed to optimal continuous sizing of gates within each block and usage of the dynamic programming heuristic for combining the various individually optimized blocks together.

For shorter delay constraints, synthesis excels over the composite curve and is even able to meet timing requirements beyond that achievable using the optimizer. The shortcoming of the heuristic lies in the conditions that were made in order to optimize these blocks separately. These conditions are illustrated in Figure 6-14.

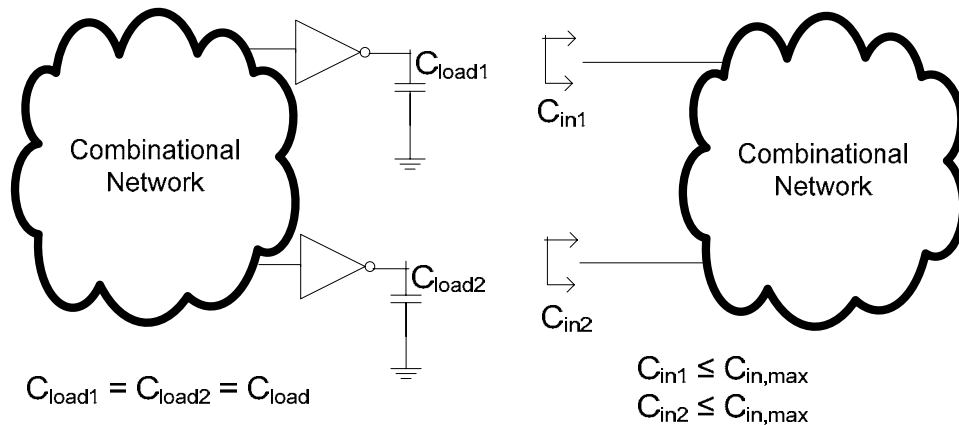


Figure 6-14. Illustration of capacitance constraints placed on inputs and outputs of circuit blocks.

In order to size the blocks separately, a fixed capacitance of  $C_{load}$  is assigned to every output of the block and the combinational network is sized to drive this load. On the inputs, a maximum input capacitance,  $C_{in,max}$ , inequality is used to limit the maximum capacitance at each input. The optimizer is free to size the circuit as long as this inequality is satisfied. Since neither of the inputs might be at  $C_{in,max}$  the driving block is usually over sizing the circuit to meet delay requirements. The optimizer is thus wasting a lot of energy trying to drive capacitances that are usually lower than the constraints. The energy overhead from upsizing is exacerbated at high performance requirements as is illustrated in Figure 6-9.

The main advantage of this heuristic is that it produces a continuous energy-delay tradeoff curve which is an invaluable tool in the investigation of circuit architectures, as demonstrated by [Markovic04]. A composite energy-delay curve can be generated rapidly to evaluate the potential savings of an architectural change, as long as all the underlying blocks have been characterized.

Results presented in this chapter demonstrate robustness of the synthesized circuit optimizer developed in this work in optimizing a large variety of circuits. By taking advantage of the isolation provided by pipeline registers, the optimizer is able to optimize large circuits such as the 53-bit multiplier with reasonable runtimes. The optimizer is also able to optimize circuits with shallow logic depth such as the 55-bit incrementer. Finally, the optimizer demonstrates its versatility by producing satisfactory results for general synthesized logic such as the exponent datapath. The reduction in energy consumption evidenced in case studies covered in this chapter demonstrates the potential energy savings that can be gained by performing energy-delay optimization using local fit parameters. Furthermore, we have developed a heuristic for combining energy-delay tradeoff curves of individual blocks for rapid architectural exploration. Energy-delay tradeoff curves generated by this heuristic are on par with results of commercial synthesis.

## 7.0 Conclusion

This thesis addresses the topic of energy-performance optimization of standard cell synthesized digital integrated circuits. Formulating this optimization problem as a geometric program provides a systematic method for arriving at the globally optimal solution. The main contributions of this thesis are:

- improvement in accuracy of analytical models employed in the geometric program by introducing localized models which are fitted to the current operating environment of the gate, such as gate size, load capacitance, and input slope;
- development of an iterative algorithm for solving the geometric program using continuously-refined local fit parameters;
- verification of the quality of the result by comparing both the continuous solution and the rounded solution with results from commercial synthesis;
- significant improvement in computation time of geometric programs of sequential circuits by modeling flip-flops as buffers that present similar loads to the fan-in, regardless of flip-flop sizing;

The optimization framework is applied on several circuits:

- a 16-bit multiplier implemented using either SVT or HVT transistors. This example outlines the need for selecting the right process technology for a design in order to minimize energy for a required performance;
- building blocks of a double-precision FPU. The optimizer produces either equal or better results compared to commercial gate sizing for circuits of various complexities;

- normalization and rounding logic. This example demonstrates how energy-delay tradeoff curves of basic building blocks can be combined together to produce a composite energy-delay tradeoff curve of the full architecture.

## 7.1 Future Work

Further work is suggested to improve the energy and delay models of the standard cells to account for internal energy due to switched internal capacitance. These internal capacitances which are not currently modeled arise in complex gates like full-adders and multiplexors. The main issue with these capacitances is that they do not usually scale linearly with gate sizing. Further work should also emphasize on modeling the dependence of delay and energy on supply ( $V_{DD}$ ) and threshold voltage ( $V_{TH}$ ).

Furthermore, with the decreasing ratio between  $V_{DD}$  and  $V_{TH}$ , resulting in increased sensitivity of propagation delay and energy on these parameters, these values should not be taken as fixed parameters from the standard cell provider but should be treated as optimization variables. The existing optimization framework can be extended to include these extra parameters using a similar scheme of performing a first-pass optimization using global fit parameters and iteratively refining the result using local fit parameters.

More work is also required to produce a better quality discrete sized solution from the continuous sized solution. This is required to close the loop in creating a standard cell digital integrated circuit optimizer because standard cells are usually only available in discrete sizes. This body of work can either extend on the continuous solution guided dynamic programming approach introduced by [Hu07] or it can also borrow ideas from

the field of operations research. An alternative to the rounding step is to map the optimized netlist to a standard cell with continuous sizes such as [Hashimoto04].

## Bibliography

- [Bhatnagar02] H. Bhatnagar: Advanced ASIC Chip Synthesis: Using Synopsys Design Compiler, Physical Compiler, and PrimeTime, Springer 2002.
- [Boyd05] S. Boyd, S.J. Kim, D.D. Patil, M. A. Horowitz, "Digital Circuit Optimization via Geometric Programming," Operations Research, pp 899-932, Nov. 2005.
- [Boyd07] S. Boyd, S. J. Kim, L. Vandenberghe, A. Hassibi, "A tutorial on geometric programming," Optimization and Engineering, pp 67-127, March 2007.
- [Chan90] P.K. Chan, "Algorithms for library-specific sizing of combinational logic," Proceedings of the 1990 Design Automation Conference, pp 353-356, August 2005.
- [Chinnery05] D. Chinnery, K. Keutzer, "Linear Programming for Sizing,  $V_{th}$ , and  $V_{dd}$  Assignment," Proceedings of ISLPED 2005, pp 149-154, June 1990.
- [Chinnery06] D. Chinnery, "Low Power Design Automation," Ph. D. Dissertation, 2006.
- [Chuang95] W. Chuang, S. Sapatnekar, "Timing and Area Optimization for Standard-Cell VLSI Circuit Design," IEEE Transactions on CAD of Integrated Circuits and Systems, vol. 14, pp 308-320, March 1995.
- [Conn99] A.R. Conn, I.M. Elfadel, W.W. Molzen Jr., P.R. O'Brien, P.N. Strenski, C. Visweswariah, C.B. Whan, "Gradient - Based Optimization of Custom Circuits Using a Static Timing Formulation," Proceedings of the 1999 Design Automation Conference, June 1999, pp 452 – 459.

- [Coudert96] O. Coudert, "Gate Sizing: a General Purpose Optimization Approach," Proceedings of the 1996 European Design and Test Conference, pp 214-218, 1996.
- [Curran06] B. Curran, et al, "4GHz+ Low-Latency Fixed-Point and Binary Floating-Point Execution Units for the POWER6 Processor," Proceedings of ISSCC 2006, pp. 436-437, Feb. 2006.
- [Fishburn85] J. P. Fishburn, A. E. Dunlop, "TILOS: a Posynomial Programming Approach to Transistor Sizing", Proceedings of ICCAD 1985, pp 326-328, Nov. 1985.
- [Hashimoto04] M. Hashimoto, K. Fujimori, H. Onodera, "Automatic Generation of Standard Cell Library in VDSM Technologies," Proceedings of ISQED 2004, pp 36-41, 2004.
- [Hu07] S. Hu, M. Ketkar, J. Hu, "Gate Sizing for Cell Library-Based Designs," Proceedings of the 2007 Design Automation Conference, pp 847-852, June 2007.
- [IEEE85] IEEE, "IEEE Standard for binary floating-point arithmetic," ANSI/IEEE Std 754-1985, 1985.
- [Kao06] S. Kao, R. Zlatanovici, B. Nikolic, "A 250ps 64-bit Carry-Lookahead Adder in 90nm CMOS," Proceedings of ISSCC 2006, pp. 438-439, Feb. 2006.
- [Keshavarzi00] A. Keshavarzi, K. Roy, C. Hawkins, "Intrinsic Leakage in Deep Submicron CMOS ICs – Measurement-Based Test Solutions," IEEE Transactions on VLSI Systems, pp 717-723, Dec 2000.

- [Leiserson91] C.E. Leiserson, J.B. Saxe: "Retiming synchronous circuitry," *Algorithmica* 6(1) pp. 5-35.
- [Li04] X. Li, P. Gopalakrishnan, Y. Xu, L. Pileggi, "Robust Analog/RF Circuit Design with Projection-Based Posynomial Modeling," *Proceedings of ICCAD 2004*, pp. 855-862.
- [Lin90] S. Lin, M. Marek-Sadowska, E.S. Kuh, "Delay and area optimization in standard-cell design," *Proceedings of the 1990 Design Automation Conference*, pp 349-352, June 1990.
- [Liu94] D. Liu, C. Svensson, "Power Consumption Estimation in CMOS VLSI Chips," *IEEE Journal of Solid State Circuits*, vol. 29, pp. 663-670, June 1994.
- [Markovic04] D. Markovic, V. Stojanovic, B. Nikolic, M. Horowitz, R. Brodersen, "Methods for True Energy-Performance Optimization," *IEEE Journal of Solid State Circuits*, vol. 39, pp 1282-1293, Aug. 2004.
- [Mosek07] Mosek Optimization Toolbox 5.0, online documentation at [www.mosek.com](http://www.mosek.com)
- [Nguyen03] D. Nguyen, et al, "Minimization of Dynamic and Static Power Through Joint Assignment of Threshold Voltages and Sizing Optimization," *Proceedings of ISLPED 2003*, pp. 158-163, August 2003.
- [Orshansky05] M. Mani, A. Devgan, M. Orshansky, "An Efficient Algorithm for Statistical Minimization of Total Power under Timing Yield Constraints," *Proceedings of DAC 2005*, pp. 309-314, June 2005.

- [Pang06] L.-T. Pang, B. Nikolic, "Impact of Layout on 90nm CMOS Process Parameter Fluctuations," VLSI 2006 Digest of Technical Papers, pp. 69-70, 2006.
- [Rabaey03] J.M Rabaey, A. Chandrakasan, B. Nikolic: Digital Integrated Circuits: A Design Perspective, 2nd edition, Prentice-Hall 2003.
- [Ruehli77] A. E. Ruehli, P. K. Wolff, G. Goertzel, "Analytical Power/Timing Optimization Technique for Digital System," Proceedings of the 1997 Design Automation Conference, pp 142-146, June 1977.
- [Sutherland99] I. Sutherland, R. Sproul, D. Harris: Logical Effort, Morgan-Kaufmann, 1999
- [Tavel06] B. Tavel, et al, "65nm LP/GP mix low cost platform for multi-media wireless and consumer applications," Proceedings of ESSDERC 2005, pp 423-426, Sept. 2005.
- [Zlatanovici06] R. Zlatanovici, "Power - Performance Optimization for Digital Circuits," Ph. D. Dissertation, 2006.