Optimal Design of Macrocells for Low Power and High Speed

Piyush K. Sancheti and Sachin S. Sapatnekar

Department of Electrical and Computer Engineering

201 Coover Hall, Iowa State University

Ames, IA 50011.

1 Introduction

With the emergence of portable products as major players in the electronics market, controlling the power dissipation of integrated circuits is gaining increased importance. While progress is being made in improved battery technology to supply a larger amount of power per unit weight of the battery, it must also be coupled with an accompanying reduction in the power dissipation of IC's. Even for nonportable applications, as the system complexity increases, it is becoming more and more important to limit the power dissipation so as to avoid additional costs for cooling down electronic systems. In this context, it has become increasingly important to design CMOS digital circuits to ensure a low power dissipation. At the same time, however, it is also necessary to ensure that the speed of the circuit is not unduly sacrificed. An additional consideration is the need for fast system turnaround times, which necessitates the use of semicustom design styles. In this work, we address a design style using flexible macrocells [1], where a library of basic functional elements, or macrocells, is constructed.

In this work, we present an accurate way of assessing the power dissipation and delay of a macrocell as a function of its transistor sizes, and use an interface with SPICE to ensure that the delay and power calculations are precise. The use of SPICE ensures that power and delay measurements can be made highly accurate using high order models. Moreover, effects such as the variations in parasitic capacitances with voltage, channel length modulation and the body effect, to name just a few are measured accurately; these effects are completely ignored by coarser models. This is very important since a macrocell is designed once and only once. At the end of this procedure, an accurate measure of the worst-case delay and worst-case power dissipation of the macrocell is available.

Any optimization algorithm entails a tradeoff between tractability, in terms of the accuracy of the model used, and accuracy. In the optimization segment, we sacrifice some accuracy in the interests of tractability. However, our optimization technique utilizes more accurate modeling functions than the conventional constant-resistor-constant-capacitor models that are often used. The novel modeling approach here uses posynomials [2] of *arbitrary* degree, thereby allowing for high accuracy. This is particularly so since low degree posynomials are already commonly used to estimate power and delay with an error of less than 20%, and therefore, our technique will do no worse than any such method.

The optimization problem is formulated as a convex programming problem and an efficient algorithm [3] is used to solve the problem. The optimization technique devised here is powerful and has potential applications in other areas where the objective and constraints are "nearly posynomial."

The chief contributions of this work are as follows. Firstly, we present a new method for characterizing the power contribution of a *single* cell to a circuit. This is very useful in getting good and fast estimates of the power dissipation of a circuit built from a library of characterized cells. Secondly, we show that the optimization problem presented here can be solved using *precise* delay and power modeling using SPICE, unlike other approaches that are restricted to more approximate models, such as Elmore delays or models that do not consider the variations of MOS capacitances with voltage. It is well-known that accuracy is very important during the one-time effort of designing cells for a library, since such cells will be used repeatedly. Thirdly, although the optimization approach presented herein is applied to the problem at hand, it is a technique that is being demonstrated. This technique may be used in problems where great modeling accuracy is important, and the objective and constraint functions can be represented as "nearly posynomial" functions.

The paper is organized as follows. Section 2 describes the idea of a flexible macrocell, on which our methods are applied. Section 3 presents the mathematical background required to understand the method. In Section 4, we explain how the area, delay and power are measured, the last two using circuit simulation for the greatest accuracy. The optimization problem is formally formulated in Section 5, and the efficient convex programming algorithm is briefly described therein. Experimental results are provided in Section 6, followed by concluding remarks in the last section.

2 Flexible Macrocells

Unlike conventional standard cell design systems, where the height of each cell is constrained to be constant, the flexible macrocell [1] idea imposes no such limitation, and provides an alternative approach to layout synthesis that can potentially make better utilization of the layout area. A schematic of a flexible cell is shown in Figure 1(a), and the arrangement of such macrocells in a layout is shown in Figure 1(b). Note that the placement of power/ground lines in the center of the cell ensures that they can be run as straight lines through a row; the approach where power/ground lines run at the top and bottom of a cell cannot ensure that these lines will be straight if the cells are of variable height. The flexible macrocell approach has been exercised in [1], where a layout synthesis system for flexible cells is described. Further CAD research for this design style is presented in [4], which describes an over-the-cell channel router for triple metal routing of systems using the flexible cell approach.

Apart from the potential for better area utilization and better performance obtainable by utilizing larger cells only where necessary, another significant advantage of using variable height macrocells lies in the fact that if the power dissipation of the cell is to be controlled, the n- and p-type transistors should not be made too large. Large transistors may succeed in delay reduction, but would lead to larger capacitance values to be charged and discharged, thereby leading to increased power dissipation.

The flexible macrocells are used as building blocks to construct large circuits, and therefore, it is critical that each individual macrocell must be well optimized for power and timing. The problem is, however, a difficult one since the performance of a flexible macrocell is dependent on the context in which it is placed in the circuit, i.e., on the fanout gates that it must drive. A library of flexible macrocells should contain several versions of the same gate, with varying driving powers, and each macrocell fully optimized. Once a library has been designed, one of several techniques such as [5, 6] may be used to select the appropriate cell optimally in a given circuit.

To date, there have been few methods that address the problem of designing flexible macrocells for a library. The most significant related work is that in [7], where a methodology for minimizing the area-delay product for designing standard cells of constant height was described. No power considerations were incorporated in that work, and the procedure used simple first-order models to calculate gate delays.

3 Mathematical Background

Definition 1 *Convex set*: A set $C \in \mathbf{R}^n$ is said to be convex if for every $\mathbf{x}_1, \mathbf{x}_2 \in C$ and every real number $\alpha \in (0,1)$, the point $\alpha\mathbf{x}_1 + (1-\alpha)\mathbf{x}_2 \in C$. Geometrically, convexity implies that given any two points $\mathbf{x}_1, \mathbf{x}_2 \in C$ all points on the line joining them lie within C.

Definition 2 *Convex function*: A function $f(\mathbf{x}) : \mathbf{R}^n \rightarrow \mathbf{R}$ is said to be convex if for any pair of points $\mathbf{x}_1, \mathbf{x}_2 \in \mathbf{R}^n$ and for every real number $\alpha, 0 \le \alpha \le 1$,

$$f[\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2] \le \lambda f(\mathbf{x}_1) + (1-\lambda) f(\mathbf{x}_2). \tag{1}$$

Geometrically, this implies that the segment joining the points lies entirely above or on the graph of $f(\mathbf{x})$.

Definition 3 *Posynomials*: A posynomial is a function g of a positive variable $\mathbf{x} \in \mathbf{R}^{\mathrm{n}}$ that has the form

$$g(\mathbf{x}) = \sum_j \gamma_j \prod_{i=1}^{n} x_i^{\alpha_{ij}} \tag{2}$$

where the exponents $\alpha_{ij} \in \mathbf{R}$ and the coefficients $\gamma_j > 0$. Roughly speaking, a posynomial is a function that is similar to a polynomial, except that

- The coefficients γ_j must be positive.

- An exponent α_{ij} could be any real number, and not necessarily a positive integer, unlike the case of polynomials.

A posynomial has the useful property that it can be mapped onto a convex function through an elementary variable transformation, $(x_i) = (e^{z_i})$ [2].

Definition 4 *Convex Programming Problem*: A convex programming problem is stated as follows:

$$\text{minimize} \quad f(\mathbf{x}) \text{ subject to } \quad \mathbf{x} \in C$$

where $f(\mathbf{x})$ is a convex function over a convex set, C. This problem has the property that any local minimum of f over S is a global minimum.

Definition 5 *Polytope*: A (convex) polytope P in an n-dimensional space is an intersection of m half-spaces, and is defined as

$$P = \{\mathbf{x} \in \mathbf{R}^n \mid A\mathbf{x} \ge \mathbf{b}\} \tag{3}$$

where $A \in \mathbf{R}^{m \times n}$ and $\mathbf{b} \in \mathbf{R}^m$.

4 Performance Modeling of Macrocells

4.1 Delay and Power Modeling

Since a macrocell in a library is designed only once, it is essential that the solution obtained by the design algorithm be optimal. As a consequence, it is imperative that accurate models be used for the delay and power dissipation of the cell. Notice that effects such as the variations in parasitic capacitances with voltage, channel length modulation and the body effect, etc. are accurately measured in this approach.

Of all the techniques currently used to simulate a circuit, circuit-level simulation provides the highest degree of accuracy. For large circuits it entails high computational costs and extensive memory requirements, but since our technique involves sizing transistors for flexible macrocells, each of which typically has less than 20 transistors, the use of circuit level simulation is justified. This statement is borne out by the CPU times of our algorithm.

We now describe a method for calculation of the power dissipation of a cell using the voltages and currents obtained from SPICE. Next, we explain how SPICE is employed to estimate the rise and fall delays for a cell in order to evaluate the propagation delay of the cell. In the modeling phase, we assume that the contributions of source/drain capacitances of transistors to the fanout capacitance of a gate can be neglected in comparison with wiring and fanout gate capacitances.

4.1.1 Power Measurement

The power dissipation of a cell is dependent on the context within which it is placed in a circuit. In this work, we present a systematic method of individually determining the contribution of each cell to the power dissipation of a circuit. To our knowledge, no work on estimating individual cell contributions to the circuit power dissipation has been published before; the approach in [8], for example, only calculates the power dissipation for an entire circuit, rather than the split-up of the power dissipation components, as calculated here.

The power dissipation of a flexible macrocell as a function of the sizes of transistors in the cell is composed of two components: the *dynamic power* and the *short-circuit power*. The power associated with the leakage current is negligible and is not considered. For an accurate estimate of these two components of power we use SPICE to monitor the voltage at nodes of interest and current in the branches of interest.

We explain the procedure for calculating the driving power for a cell by means of an example of an inverter that is being driven by another inverter, *Instage* and has a fanout of a certain number minimum size inverters (corresponding to the driving power that the cell is being designed for) that together form *Outstage*, as shown in Figure 2. INV is the gate that is being sized.

In case of multiinput cells, the gate terminals of all the transistors are driven by the same inverter in *Instage*. As a result, during a transition, all of the pMOS or all of the nMOS transistors in the cell switch and hence, the maximum power is drawn by the cell, since all of the gate terminal capacitances of the cell must be charged. Moreover, the capacitance driven by *Instage* is the maximum, which ensures the most pessimistic estimate of the transition time at each input, and correspondingly, pessimistic calculations of the short circuit dissipation. This ensures that the power dissipation estimated in our approach is in fact the worst case power dissipation for the cell. Note that in the succeeding discussion, we use the term “power” to describe the power dissipated by the cell in each transition, W.

If the period of the clock is T, then assume that voltage waveform at node 4 is high for the interval $[0, T/2]$ and low for the interval $[T/2, T]$. As a consequence the voltage waveform at node 6 will be low for $[0, T/2]$ and high for $[T/2, T]$.

To find out the current through a branch, we insert 0V independent voltage source in that branch and then compute (via circuit simulation) the current through the voltage source. Four such sources, $V_{s1} \cdots V_{s4}$ are inserted in the circuit. We now explain the procedure for calculation of dynamic and short circuit power by means of the inverter example shown in Figure 2; as mentioned earlier, INV is the gate being sized.

(1) **Calculating the dynamic power:** Notice first that the components of the dynamic power (or CV^2 power) that depend on transistor sizes in INV are caused by the current required to drive the following (all other components of the dynamic power dissipation are independent of the transistor sizes of interest):

(a) *the gate terminal capacitances of transistors in INV.*

This component of the dynamic power is measured by monitoring voltage at node 3 and the current through voltage source V_{s1}.

For the interval $[0, T/2]$, there is a path from V_{DD} to node 3, and hence the dynamic power is given by the product $[(V_{DD} - V(3)) \cdot i(V_{s1})]$ numerically integrated over time using the trapezoidal formula [9]. For the interval $[T/2, T]$, there is a path from node 3 to ground and hence the dynamic power is the product of $V(3)$ and $i(V_{s1})$ integrated over time.

(b) *the source/drain terminal capacitances of transistors in INV.*

This component of the dynamic power is measured by monitoring voltage at node 6 and the current through the lumped capacitor C, given by $i(V_{s3}) - i(V_{s2}) - i(V_{s4})$. The power computation is similar to that in (a) above. Note that C corresponds to the source/drain capacitances of transistors in the cell, and that the gate capacitance of the fanout gates are at node 8.

(2) **Calculating the short-circuit power:** The only component of the short-circuit power that depends on transistor sizes in INV is the power due to the V_{DD}-to-GND current through INV.

To calculate the short-circuit power, we monitor the transistor that is off during that half cycle. The current through this transistor is the short circuit current, since the other transistor carries not only the short circuit current, but also the dynamic current required to charge the capacitances at the output. For the interval $[0, T/2]$, the pMOS transistor in the cell is off (since the output of the cell at node 6 is low), and hence the short circuit power for this half cycle is the product of V_{DD} and the current, $i(V_{s2})$ integrated over time. Similarly, for the interval $[T/2, T]$, the nMOS transistor in the cell is off, and hence the short circuit power is the product of V_{DD} and the current, $i(V_{s3})$ integrated over time. The total power required to drive the inverter cell is the sum of the dynamic and the short-circuit power.

Note that the total power dissipation of the circuit in Figure 2, P_{tot}, is related to the power dissipation of the flexible macrocell, P_{cell}, by the relation $P_{tot} = (P_{cell} + \text{constant})$, where the constant term consists of power dissipation components that are independent of transistor sizes in the cell. Hence if the objective were to minimize the power dissipation of the cell, one could simply minimize P_{tot}. However, if one wants to perform a constrained optimization with specifications on P_{cell}, as we are doing here, it is important to estimate P_{cell} accurately.

4.1.2 Delay Measurements

We define the transition delay of a gate as the amount of time required by its output waveform to cross the 50% threshold, after its input waveform has crossed its 50% threshold. For the purposes of delay calculation, for each gate, we assume that only one input is switching (note that this differs from the assumption for power calculation).

Assuming that the capacitance at the output node is much larger than those at any nodes within the cell, the worst-case rise (fall) delay occurs under the condition when the largest resistance path between the output and V_{DD} (GND) is activated. The rise delay calculation procedure hence consists of the following steps; the fall delay calculation is analogous. The delay of the cell is taken to be the maximum of the rise and fall delays.

(1) **Identifying the maximum resistance path:** Coarsely speaking, the on-resistance of a transistor is given by the relationship $R_{on} \propto 1/x$, where x is the width of the transistor [10]. Hence the path of maximum resistance, Q_p (Q_n) in the p-transistor (n-transistor) network is the one for which the sum of $1/x$'s for the transistors lying on that path is maximum. A path enumeration using a depth first search is carried out to determine the maximum resistance path; since the number of paths is small, this can be performed very fast. Note that this algorithm for finding the worst case delay paths for the rise and fall transitions is a heuristic but is accurate enough for purposes of identifying the worst-case path. A similar approach has also been used in [10].

(2) **Calculating the worst-case delay:** Having identified Q_p and Q_n, the next step is to evaluate the delays associated with these paths using SPICE. To calculate the rise (fall) delay, we set the

voltages at the gate terminals of all transistors lying on Q_p (Q_n) to 0V (V_{DD}), except for the transistor that is closest to the power supply, identified as M. All other transistors in the p(n) transistor network are forced off. The input to M is switched from V_{DD} to 0V (0V to V_{DD}), so that the transistor switches on. Since all other transistors along Q_p (Q_n) are already on, the output of the cell, undergoes a transition. Using this set of inputs, SPICE is used to calculate the transition delay by subtracting the 50% thresholds of the output and input waveforms.

For example, in the circuit in Figure 3, where the number alongside each transistor denotes its width, the path A-B-C is the maximum resistance path. The transistor with input C corresponds to M. For worst-case timing purposes, therefore, a circuit simulation is carried out with these transistors being turned on and all other transistors being turned off.

4.2 Area Modeling

The area model that we use in this work is presented in this section. Note that although the area is not one of the parameters in the constraint or the objective here, the height of the cell is a constraint; the cell height is intimately linked with the area model. The layout of the cell for this model has vertical polysilicon wires running over horizontal diffusion strips, as shown in Figure 1. Hence, the height of a cell is a function of the maximum transistor width in the cell, while the width is a function of only the number of transistors in the cell. The approach here assumes that the flexible macrocells are fully complementary CMOS gates.

The following design rules have been assumed for the design of cells.
minimum line width = 2λ; minimum transistor width = 2λ; $L = 2\lambda$;
poly-to-contact spacing = 2λ; ndiff-to-pdiff spacing = 10λ; contact size = $4\lambda \times 4\lambda$.
Based on these design rules and the fixed layout style, area models for various cells may be developed. For example, for an N-input NAND gate, it can easily be shown that

$$\begin{aligned} Cell\ Width &= (N \cdot 10 + 10) \cdot \lambda && (4) \\ Cell\ Height &= (\max_{1 \le i \le N} [W_p(i) + W_n(i)] + 10)\ \lambda && (5) \end{aligned}$$

where $W_p(i)$ and $W_n(i)$, are, respectively, the width of the pMOS and nMOS transistors connected to the i^{th} input. The area of the cell is given by $Area = (Cell\ Width) \cdot (Cell\ Height)$.

Notice that the area of a cell is the maximum of posynomial functions of transistor widths in the cell, and hence maps on to a maximum of convex functions, a convex function. The approach is not restricted to the assumed layout style; rather, any regular layout style where the area function can be presented as a maximum of posynomials can be supported. This method can also be used for area minimization under delay and/or power constraints.

In addition to the above constraints, one may wish to ensure that the height of no cell exceeds a given number (such a number can be arrived at, for example, using the technique in [7].) When the performance constraints are too tight, and can be achieved only by overshooting the height constraint, we resort to folding transistor gates to satisfy the cell height constraints, and repeat the optimization under a new area model.

5 The Optimization Algorithm

5.1 The Convex Program Formulation

The optimization problem is stated as follows:

$$\begin{array}{rll} \text{minimize} & Power(\mathbf{x}) & \\ \text{subject to} & Delay(\mathbf{x}) & \leq D_{spec} \\ & Height & \leq H_{spec} \end{array} \tag{6}$$

where $\mathbf{x}$ is the vector of transistor sizes within the cell.

The rationale for formulating the optimization problem as a convex program is explained in this section. It is apparent from Definition 3 that under the transformation, $x_i = e^{z_i}$, the posynomial height function becomes convex.

The delay of a cell is well-approximated (to about 10-20%) by the Elmore delay [11], which is a posynomial function of the transistor sizes. This is a low-degree posynomial in which the exponents of the terms are either -1, 0 or 1. Both the short circuit and the dynamic power dissipation of a circuit can also be expressed as low-degree posynomial functions of the transistor sizes [12]. This power model assumes the parasitic capacitances to be constant, which is not true in practice since gate capacitance vary with biasing; however, it does not give grossly inaccurate power estimates.

Therefore, since low degree posynomials are capable of providing good approximations to the delay and power functions, and posynomials are a versatile class of functions, it is very likely that the use of higher degree posynomial functional approximations will provide much more accurate models of the delay and power dissipation of a flexible macrocell. We now connive to formulate the problem in such a way that posynomials of *arbitrarily high degree* are used to model these two quantities, using the following strategy. The process is illustrated in Figure 4. Note that since the real power(delay) function is "almost" a posynomial function, the real feasible region is "almost" convex. Allowances for slight nonconvexities can be made as in [13].

Assuming that the delay and the power, which determine the feasible region of the optimization problem, can be approximated by posynomial functions (of arbitrary degree), the transformation $x_i = e^{z_i}$ will map the feasible region onto a convex set in the $\mathbf{z}$ space. The optimization problem is then one of minimizing a convex function, the power, over this convex set in the $\mathbf{z}$ space.

An important characteristic of the optimization algorithm that we employ is that it does not require the constraints describing the feasible set to be enumerated, but merely requires feasibility checks for a given point, and gradient evaluations. Thus, the beauty of this optimization strategy is that we may use implicitly posynomials of arbitrarily high degree, without *ever* having to explicitly enumerate the approximating functions.

Example: To see how a feasible region specified by posynomial constraints maps on to a convex set, consider the two posynomial constraints below:

$$x^2 + y^2 \leq 1 \quad \text{and} \quad xy \leq 0.2$$

Note that we only consider the region where $x, y > 0$.

The plot in Figure 5(a) shows that the feasible region is nonconvex in the $\mathbf{x}$-space; however, upon performing the transformation described above, the feasible region becomes convex in the new space. This may be seen in Figure 5(b), where both axes are plotted on a logarithmic scale.

5.2 The Convex Programming Algorithm

The optimization algorithm used here is an efficient convex programming algorithm [3]. Details of the implementation are provided in [14], and only a superficial explanation is provided here to avoid redundancy. Here, we presuppose the reader's familiarity with the algorithm in [14]. The algorithm works in an n-dimensional space, where n is the number of variables. It encloses the solution in an initial polytope (recall that a polytope was defined in Section 3), which is taken to be a box, described by

$$\mathcal{P} = \{\mathbf{z} \mid \log(x_{i,min}) \leq z_i \leq \log(x_{i,max})\} \tag{7}$$

where $x_{i,min}$ and $x_{i,max}$ are, respectively, the minimum and maximum size for each transistor.

In each iteration, another constraining half-space is added to the set of equations describing the polytope. By adding this extra half-space appropriately, the volume of the polytope is shrunk by approximately half in each iteration, until the polytope becomes sufficiently small.

The procedure for determining a hyperplane and its corresponding half-space that shrinks the polytope volume by about half is now described. In every iteration, the center of this polytope is determined by minimizing a log-barrier function, as explained in [14]. Next, we determine whether or not this center lies within the feasible region of the optimization problem. A hyperplane

$$\mathbf{c}^T\mathbf{z} = b, \quad \text{where } \mathbf{c}, \mathbf{z} \in \mathbf{R}^n, b \in \mathbf{R} \tag{8}$$

is then passed through the center of the polytope such that the solution to the problem lies in its negative half-space. The value of $\mathbf{c}$ is given by the following rule:

- If the center is feasible, $\mathbf{c} = \nabla Power(\mathbf{z})$, the gradient of the objective (power) function
- If the center is not feasible, $\mathbf{c} = \nabla g(\mathbf{z})$, the gradient of a violated (delay or height) constraint.

This hyperplane roughly halves the volume of the original polytope. The center of this polytope is then found, and the procedure is iterated until the polytope volume is sufficiently small.

5.3 Summary of the Algorithm

The overall flow of the algorithm is described in Figure 6. After reading in the circuit and the parameters, an initial polytope is created as described earlier. In each iteration, the transistor sizes are set to be the exponent of the corresponding coordinate of the polytope center. Next, the circuit is simulated, with transistor sizes set to correspond to the current polytope center, using SPICE to calculate the delay and power as described in Section 4.1. Depending on whether the delay constraint is met or not, the gradient of either the delay or the power is used to generate the new half-space that halves the polytope volume. The procedure continues until the polytope is small enough; termination criteria are described in [14].

The total number of SPICE simulations required in each iteration is $n + 2$, where n is the number of variables. Two simulations are required to evaluate the delay and power, respectively, and n more simulations are required to compute the delay or power gradient using finite differences.

The number of variables for this problem is extremely small (typically less than 20). Since each oracle invocation involves $n+2$ circuit simulations, the dominant component of the CPU time is due to simulations. In spite of this, the CPU times were seen to be reasonable since the circuit to be simulated is very small.

6 Experimental Results

The algorithm for designing flexible macrocells was implemented in a C program. The input to the program is a SPICE deck that gives a transistor-level netlist of the circuit, the delay specification, D_{spec}, and the optimal power dissipation, P_{opt}, resulting from the solution of the convex program. Both of these parameters are normalized with respect to the parameter values when all transistors in the macrocells are at minimum size. The notation used here is that the factor under the D_{spec} (P_{opt}) column in Table 1 divides (multiplies) the delay (power dissipation) of a minimum sized

Table 1: Designing standard cells for various sets of power and delay constraints

Circuit	D_{spec}	P_{opt}	Cell Height	Cell Width	Number of Iterations	Number of Simulations	CPU Time
INV	5.3x	13.3x	60		13	40	79s
	4.0x	3.2x	24	20	15	44	87s
	1.2x	1.1x	15		10	13	28s
NOR2	4.4x	9.6x	51		19	96	192s
	3.1x	2.7x	21	30	19	40	90s
	2.5x	2.1x	18		21	42	96s
NAND2	2.2x	9.6x	48		20	101	209s
	1.8x	4.4x	30	30	17	38	89s
	1.4x	1.4x	16		17	74	158s
NOR3	3.8x	9.3x	60		34	239	470s
	2.6x	2.4x	20	40	27	70	184s
	2.0x	1.7x	17		28	71	187s
NAND3	1.7x	7.4x	45		20	57	158s
	1.5x	2.2x	22	40	26	21	76s
	1.3x	1.3x	16		21	46	127s
2,2-AOI	2.2x	2.4x	21		34	115	296s
	1.9x	1.7x	17	50	35	84	230s
	1.5x	1.4x	16		34	51	168s
2,2-OAI	2.2x	3.5x	26		30	63	177s
	1.9x	1.8x	17	50	37	94	230s
	1.5x	1.4x	16		31	48	154s

inverter. For example, a 4x factor for delay implies a delay that is a quarter of that for the minimum-sized cell, and a 10x factor for power implies that the power dissipation is 10 times that for the minimum-sized cell.

The transistor sizes were constrained to be no less than 2λ in width while the transistor lengths were kept constant at the allowable minimum of 2λ. It is important to note that these parameters were chosen just to demonstrate the application of the flexible macrocell design algorithm and that any other reasonable assignment of values to the parameters is possible.

The algorithm was tested on seven different gates: INV, NAND2, NOR2, NAND3, NOR3, 2,2-AOI, and 2,2-OAI for different sets of values of D_{spec}.

Table 1 shows the height and width of each cell, the number of SPICE simulations, the number of iterations of the convex programming algorithm, and the CPU time required on an HP9000/700 workstation.

Since our method solves the underlying convex programming problem exactly, the power dissipation shown in Table 1 correspond to the globally optimum solution to the problem for that layout style, with an accuracy dictated by the user-specified termination criterion described in [14].

It was observed from our experiments that, as expected, as D_{spec} is made more stringent, the area and the power dissipation of the flexible macrocell increase. This is because as the width of a transistor increases, the current required to drive the transistor increases, which in turn means that more power is required by the source to drive it. On the other hand, increasing the width of a transistor reduces the resistance of the transistor and hence may contribute to reducing the delay.

The execution time depends largely on the number of SPICE simulation required. This is not surprising since SPICE simulations constitute the most computationally intensive step in the entire design process.

Our claim of optimality is supported by the fact that we found that in each case, the same result was obtained, regardless of the initial polytope used. The property of the convex programming algorithm is that it will always give the optimal solution when applied to a convex programming problem; if the problem is not a convex program, then different initial polytopes will lead to different results [15]. In this work, we are not guaranteed that the problem is a convex program; we merely approximate the functions by a convex region in a different space. Therefore, if the approximations did not lead to convex programs in practice, the result would differ for different initial polytopes, which was not seen to happen on any of our test cases.

7 Conclusion

A new technique for designing flexible macrocells for a library has been developed. The problem uses accurate models for estimating the power dissipation and delay of a cell, and formulates the optimization problem as a convex program. An efficient technique is then used to perform the optimization.

The contributions of this work lie primarily in the methodology that is used for power estimation and in the new optimization algorithm that uses arbitrary order posynomials for modeling the delay and power of the macrocell. For the modules that are described in our section on experimental results, the run times are seen to be small.

Although we have tried to keep our modeling efforts as accurate as possible, there are a few sources of inaccuracy, primarily associated with approximating the delay and power by posynomial functions. These are not seen to be significant in practice.

8 Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments.

References

[1] J. Kim, S. M. Kang, and S. S. Sapatnekar, “High performance CMOS macromodule layout synthesis,” in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 4.179–4.182, 1994.

[2] J. Ecker, “Geometric programming: methods, computations and applications,” *SIAM Review*, vol. 22, pp. 338–362, July 1980.

[3] P. M. Vaidya, “A new algorithm for minimizing convex functions over convex sets,” *Proc. IEEE Foundations of Computer Science*, pp. 338–343, Oct. 1989.

[4] J. Kim and S. M. Kang, “A new triple-layer OTC channel router,” in *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 647–650, 1994.

[5] P. K. Chan, “Algorithms for library-specific sizing of combinational logic,” in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 353–356, 1990.

[6] S. Lin, M. Marek-Sadowska, and E. S. Kuh, “Delay and area optimization in standard-cell design,” in *Proceedings of the ACM/IEEE Design Automation Conference*, pp. 349–352, 1990.

[7] S. M. Kang, “A design of CMOS polycells for LSI circuits,” *IEEE Transactions on Circuits and Systems*, vol. 28, pp. 837–843, Aug. 1981.

[8] S. M. Kang, “Accurate simulation of power dissipation in VLSI circuits,” *IEEE Journal of Solid-State Circuits*, vol. SC-21, pp. 889–891, Oct. 1986.

[9] A. Ralston and P. Rabinowitz, *A First Course in Numerical Analysis.* New York: McGraw-Hill, 1978.

[10] J. Fishburn and A. Dunlop, “TILOS: A posynomial programming approach to transistor sizing,” in *Proceedings of the IEEE International Conference on Computer-Aided Design*, pp. 326–328, 1985.

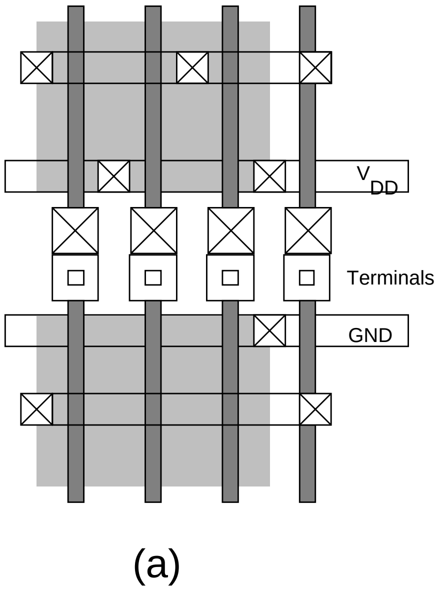
[11] J. Rubenstein, P. Penfield, and M. A. Horowitz, “Signal delay in RC tree networks,” *IEEE Transactions on Computer-Aided Design*, vol. CAD-2, pp. 202–211, July 1983.

[12] H. J. M. Veendrick, “Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits,” *IEEE Journal of Solid-State Circuits*, vol. SC-19, pp. 468–473, Aug. 1984.

[13] S. S. Sapatnekar, P. M. Vaidya, and S. M. Kang, “Convexity-based algorithms for design centering,” in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, pp. 206–209, 1993.

[14] S. S. Sapatnekar, V. B. Rao, P. M. Vaidya, and S. M. Kang, “An exact solution to the transistor sizing problem for CMOS circuits using convex optimization,” *IEEE Transactions on Computer-Aided Design*, vol. 12, pp. 1621–1634, Nov. 1993.

[15] P. Vaidya. *Private Communication*, 1993.



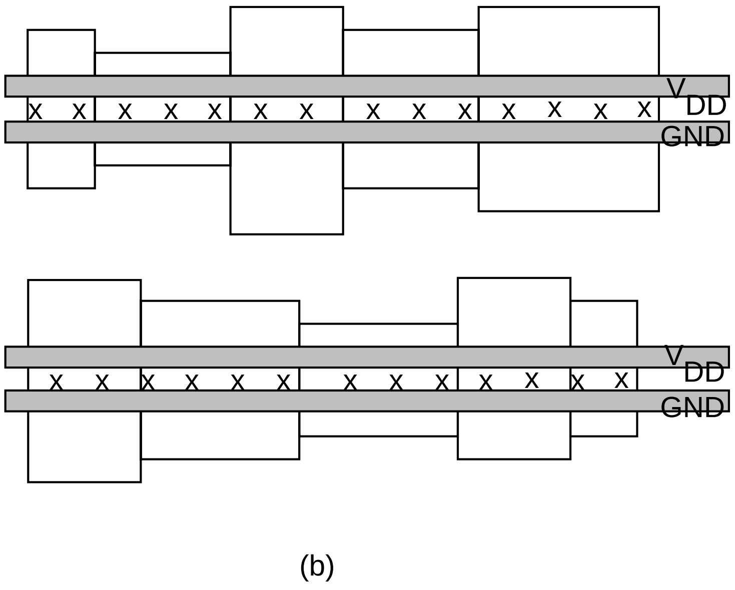


Figure 1: (a) A Flexible Macrocell (b) Rows of Macrocells in a Layout

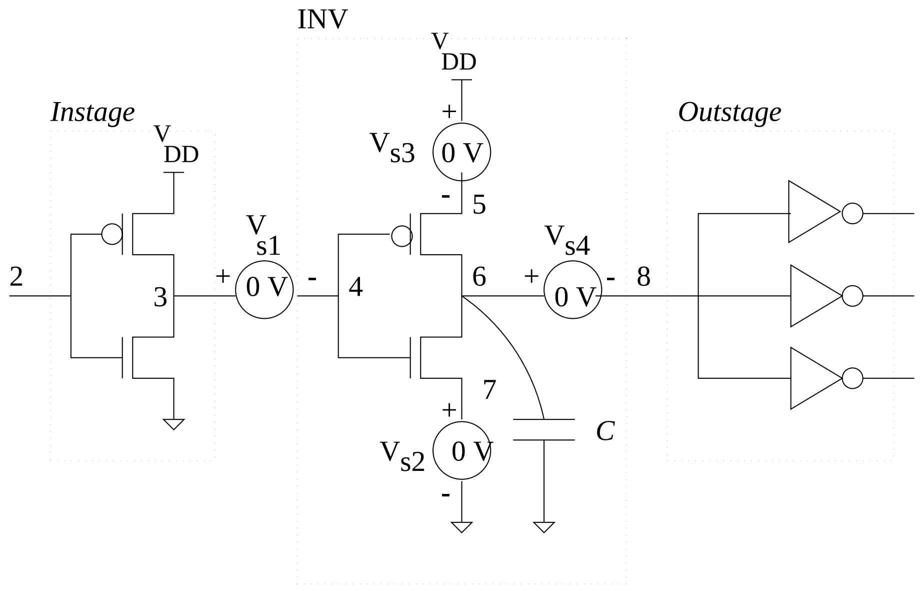


Figure 2: Example showing the procedure for calculating power dissipation

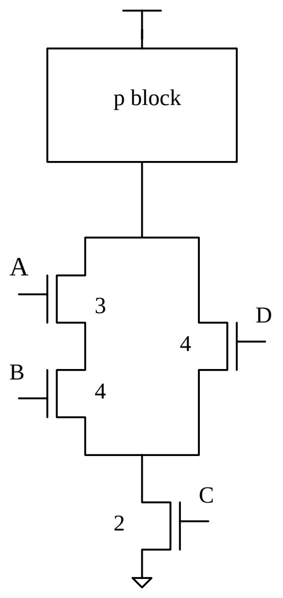


Figure 3: Calculating the worst-case delay of a cell

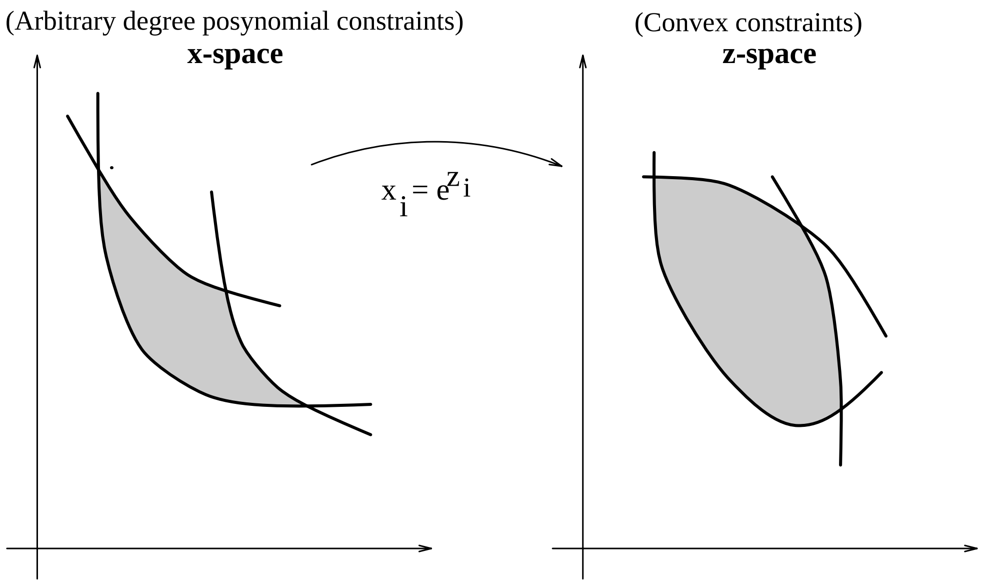


Figure 4: Explanation of the convex programming formulation

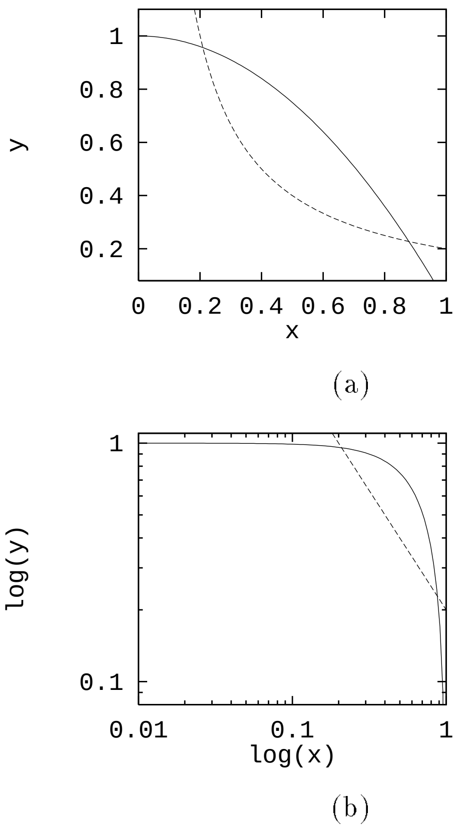


Figure 5: Example illustrating the transformation

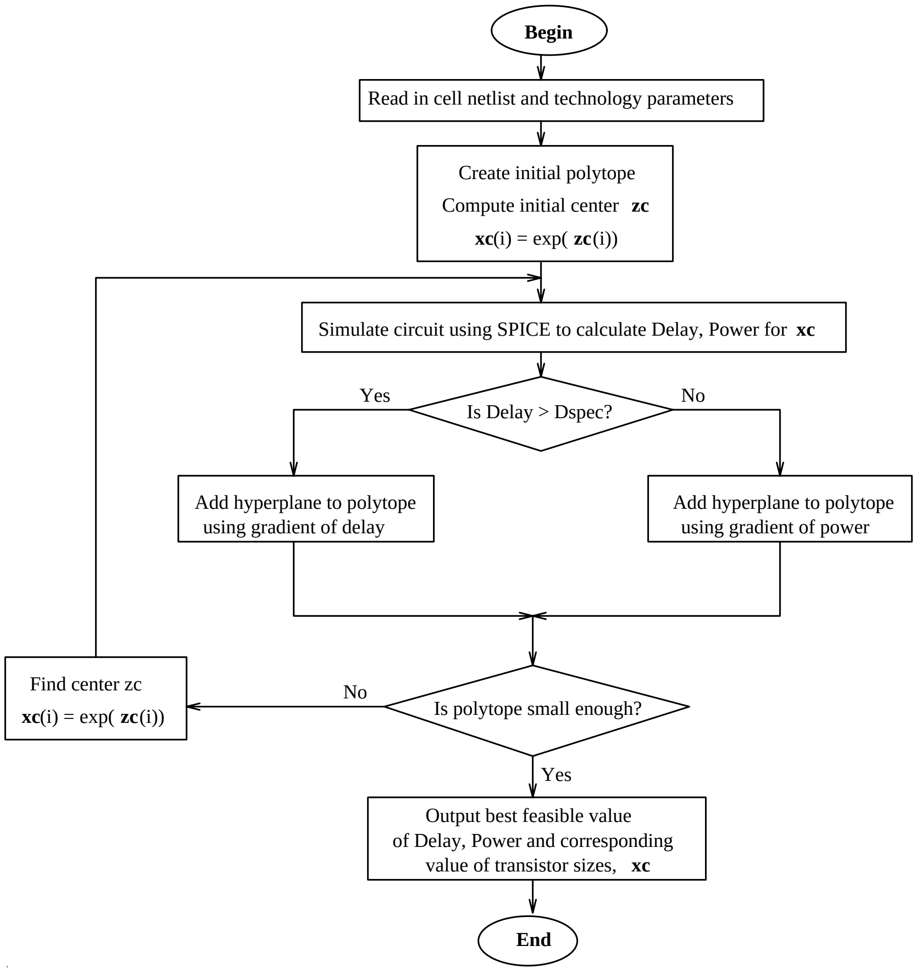

Begin
Read in cell netlist and technology parameters
Create initial polytope
Compute initial center zc
xc(i) = exp(zc(i))
Simulate circuit using SPICE to calculate Delay, Power for xc
Is Delay > Dspec?
Yes
No
Add hyperplane to polytope using gradient of delay
Add hyperplane to polytope using gradient of power
Is polytope small enough?
No
Find center zc
xc(i) = exp(zc(i))
Yes
Output best feasible value of Delay, Power and corresponding value of transistor sizes, xc
End

Figure 6: Outline of the algorithm