Layout Optimization Using Arbitrarily High Degree Posynomial Models

Piyush K. Sancheti† and Sachin S. Sapatnekar‡

† Cadence Design Systems, Inc., 270 Billerica Road, Chelmsford, MA 01801.

‡ Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011.

Abstract

The problem of designing individual macrocells for a library with power and speed considerations is addressed here. A new technique for optimization using posynomial [1] approximating functions is devised. In the design of each macrocell, optimality in design is critical and highly accurate techniques for measuring the performance are required during optimization. This paper presents methods for accurately estimating the worst-case contribution of the power and delay of a cell to a circuit. The program uses circuit-level simulation to calculate the power dissipation and delay of the cell with the highest accuracy. A rationale for using *arbitrary* degree posynomial modeling functions for area, delay and power modeling is presented. The problem is then formulated as a convex programming problem, and a rigorous optimization technique is used to arrive at the optimal macrocell.

I. Introduction

With the emergence of portable products as major market players, 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 [2], 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. Our optimization technique utilizes more accurate modeling functions than the conventional constant resistor-constant capacitor models that are often used. Effects such as the variations in parasitic capacitances with voltage, channel length modulation and the body effect, etc. are accurately measured in this approach. The novel modeling approach here uses posynomials [1] of *arbitrary* degree (without explicitly enumerating the models!), 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 is guaranteed to do no worse than any such method. The optimization problem is formulated as a convex programming problem, i.e., a problem of minimizing a convex function over a convex set. This problem has the property that any local minimum is a global minimum. The optimization methodology devised here is powerful and has potential applications in other areas where the objective and constraints are "nearly posynomial."

II. Flexible Macrocells

Unlike conventional standard cell design systems, where the height of each cell is constrained to be constant, the flexible macrocell [2, 3] 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.

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 simply to satisfy the constant-height requirements.

Since flexible macrocells are to be used as building blocks to construct large circuits, it is critical that each individual macrocell must be well optimized. 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.

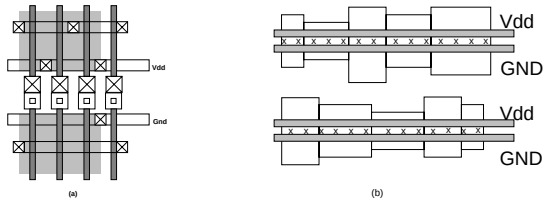

Figure 1: (a) A Flexible Macrocell (b) Macrocell Layout

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 [4], 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.

III. Performance Modeling of Macrocells

Since a macrocell in a library is designed only once, it is essential that the solution obtained by the design algorithm be optimal. Therefore, it is imperative that accurate models be used for delay/power measurements for the cell. Of all the techniques for simulating a circuit, circuit-level simulation (SPICE) provides the highest degree of accuracy. Since our technique involves sizing transistors for flexible macrocells, each of which typically has less than 20 transistors, the use of SPICE simulations does not entail high computational costs. This statement is borne out by the CPU times of our algorithm.

A. Power Measurement

The power dissipation of a cell is dependent on the context within which it is placed in a circuit. We present a systematic method of individually determining the contribution of each cell to the circuit power dissipation. To our knowledge, no work on estimating *individual* cell contributions to the circuit power dissipation has been published before.

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. We use SPICE to monitor the voltage at nodes of interest and current in the branches of interest. Note that in the succeeding discussion, we use the term "power" loosely; what is being calculated is the power per transition, W. Since, Power $= W \times f$ (where f, the switching count for the cell, is dependent on the context in which it is placed in the circuit), it is meaningful to place constraints on the power per transition, W.

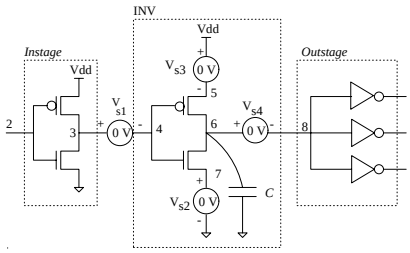

Figure 2: Calculating the power dissipation

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 min-sized 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 the short-circuit power is maximized. 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.

If the period of the clock is T, then assume that voltage at node 4 is high during $[0, T/2)$ and low during $[T/2, T)$. To find out the current through a branch, we insert a 0V independent voltage source in that branch and then compute the current through the voltage source. Four such sources, $V_{s1} \cdots V_{s4}$ are inserted in the circuit. The procedure for calculation of dynamic and short circuit power of INV in Figure 2 is as follows:

Calculating the dynamic power: The components of the dynamic power that depend on transistor sizes in INV are caused *only* by the current required to drive:
(a) *the gate terminal capacitances of transistors in INV.* 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. 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 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.

Calculating the short-circuit power: To calculate the short-circuit power, we monitor the transistor that turns off during that half cycle. The current through this transistor is the short circuit current; 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 $P_{tot} = (P_{cell} +$ 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 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 allowing under this framework, it is important to estimate P_{cell} accurately. In any case, the method shown here presents a way of characterizing the power dissipation of a macrocell.

B. 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.

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 transistor width.

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 (DFS) is carried out to determine the maximum resistance path; since the number of paths is small, this enumeration 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.

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 the transistor, M, closest to the power supply. 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.

C. Area Modeling

The area model used here is the same as that in [4] for fully complementary CMOS gates laid out as in Figure 1. Assuming the following design rules for the design of cells: (a) The minimum and maximum widths of a transistor are 2 λ and 50 λ (minimum feature size = 2λ), and (b) the length of all transistors in the cell = 2 λ, area models for various cells may be developed.

Example (Three-input NAND gate):

A transistor's contribution to the width of a cell is 10 λ. Of this, 4 λ is due to the size of the contact, 2 λ occurs because of the required contact-to-polysilicon spacing on either side; the remaining 2 λ is due to the transistor length. The cell width for this layout style is independent of the transistor widths.

$$\text{Cell Width } w = (N \cdot 10 + 10) \cdot \lambda \tag{1}$$

where the factor N represents the number of inputs to the cell. The height of a cell in this model is a function of the maximum width transistors in the cell, and is given by

$$\text{Cell Height } h = (\max_{1 \le i \le 2} [W_p(i) + W_n(i)] + 10) \ \lambda \tag{2}$$

where $W_p(i)$ and $W_n(i)$, are, respectively, the width of the pMOS and nMOS transistors connected to the i^{th} input. In each case, the 10 λ contribution is due to the spacing requirements between p- and n- type diffusion. The area of the cell, $Area = h \cdot w$. □

One may also wish to limit the height of each cell. When the performance constraints are too tight, and can be achieved only one would resort to folding transistor gates to satisfy the cell height constraints, and repeat the optimization under a new area model.

IV. The Optimization Algorithm

A. The Convex Programming Formulation

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

$$g(\mathbf{x}) = \sum_j \gamma_j \prod_{i=1}^{n} x_i^{\alpha_{ij}} \tag{3}$$

where exponents $\alpha_{ij} \in \mathbf{R}$ and coefficients $\gamma_j > 0$. Roughly speaking, a posynomial is a function that is similar to a polynomial, except that (a) coefficients γ_j must be positive, and (b) exponents α_{ij} could be a real numbers, and not necessarily a positive integer. A posynomial can be mapped onto a convex function through an elementary variable transformation, $(x_i) = (e^{z_i})$ [1].

The optimization problem may be stated as follows:

$$\text{minimize } Power(\mathbf{x}) \tag{4}$$
$$\text{such that } Delay(\mathbf{x}) \le D_{spec}; \quad Height(\mathbf{x}) \le H_{spec}$$

where $\mathbf{x}$ is the vector of transistor sizes within the cell. Alternative formulations with one of the power, area and delay being objectives and the other two providing constraints are handled equivalently.

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.[1] This method can also be used for area minimization under delay and power constraints. The delay of a cell is well-approximated (to about 10-20%) by the Elmore delay, 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. The short circuit and the dynamic power dissipation of a circuit could also be expressed as low-degree posynomial functions of the transistor sizes [5], if the parasitic capacitances were constant under biasing (this is not strictly true in practice).

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

[1] 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 posynomial functions can be supported.

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 3. 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 [6].

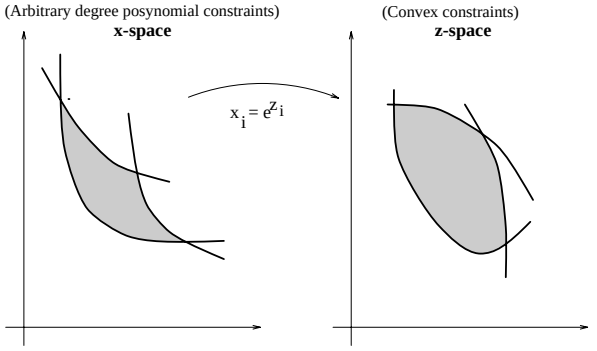

Figure 3: The transformation to convex programming.

Therefore, by approximating the delay and the power dissipation 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 now one of minimizing a convex function, the power, over this convex set in the $\mathbf{z}$ space.

We employ a convex programming algorithm described in [7]. An important characteristic of this algorithm 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.

The number of variables for this problem is extremely small. We require two circuit simulations, one each for testing whether the delay and the power constraints are violated. We have used finite differences to estimate delay and power gradients for this work. The dominant component of the CPU time is, therefore, due to simulations. In spite of this, the CPU times were seen to be reasonable since the circuit to be simulated is very small.

V. 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 power dissipation, P. Both of these parameters are normalized with respect to the parameter values when all transistors in the macrocells are min-sized. The notation used here is that the factor under the D_{spec} (P) column in Table 1 divides (multiplies) the delay (power dissipation) of a min-sized inverter. For example, a 4x factor for delay implies a delay that is a quarter of that for the min-sized cell, and a 10x factor for power implies that the power dissipation is 10 times that for the min-sized cell.

Results of the algorithm on four different gates are shown here: INV, NAND2, NOR3, and 2,2-AOI, for different values of D_{spec}. Table 1 shows the height h and width w of each cell, the number of SPICE simulations, the number of iterations of the convex programming algorithm, and the CPU time on an HP715 workstation.

Table 1: Results for various power and delay constraints

Circuit	D_{spec}	P	h	w	# Iter.	# Sim.	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	28
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
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

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 that is dictated by the user-specified termination criterion [7].

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. However, the runtimes are seen to be acceptable.

References

[1] J. Ecker, "Geometric programming: methods, computations and applications," *SIAM Rev.*, vol. 22, pp. 338–362, July 1980.

[2] J. Kim, S. M. Kang, and S. S. Sapatnekar, "High performance CMOS macromodule layout synthesis," in *Proc. ISCAS*, pp. 4.179–4.182, 1994.

[3] J. Kim and S. M. Kang, "A new triple-layer OTC channel router," in *Proc. CICC*, pp. 647–650, 1994.

[4] S. M. Kang, "A design of CMOS polycells for LSI circuits," *IEEE Trans. Circuits and Syst.*, vol. 28, pp. 837–843, Aug. 1981.

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

[6] S. S. Sapatnekar, P. M. Vaidya, and S. M. Kang, "Convexity-based algorithms for design centering," in *Proc. ICCAD*, pp. 206–209, 1993.

[7] 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 Trans. Computer-Aided Design*, vol. 12, pp. 1621–1634, Nov. 1993.