11-1-1993

ASAP: A Transistor Sizing Tool for Speed, Area, and Power Optimization of Static CMOS Circuits

Santanu Dutta
Princeton University, Electrical Engineering

Sudip Nag
Carnegie-Mellon University, Electrical Engineering

Kaushik Roy
Purdue University School of Electrical Engineering

ASAP: A TRANSISTOR SIZING TOOL FOR SPEED, AREA AND POWER OPTIMIZATION OF STATIC CMOS CIRCUITS

SANTANU DUTTA
SUDIP NAG
KAUSHIK ROY

TR-EE 93-44
NOVEMBER **1993**

SCHOOL OF ELECTRICAL ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, INDIANA 47907-1285

ASAP: A Transistor Sizing Tool for Speed, Area, and Power Optimization of Static CMOS Circuits

Santanu Dutta
Electrical Engineering
Princeton University

Sudip Nag
Electrical Engineering
Carnegie-Mellon University

Kaushik Roy
Electrical Engineering
Purdue University

Category : Computer-Aided Design (Circuit Optimization)

Please direct all correspondence to :
Kaushik Roy
Electrical Engineering
Purdue University
West Lafayette, IN 47907-1285, USA
Ph: 317-494-2361
FAX: 317-494-6440
e-mail: kaushik@ecn.purdue.edu

ASAP: A Transistor Sizing Tool for Speed, Area, and Power Optimization of Static CMOS Circuits

Abstract

This paper introduces an automated transistor sizing tool (ASAP) that incorporates accurate gate-level functional models and can be used for delay, area, and power optimization of CMOS combinational logic circuits in a VLSI design environment. ASAP considers the performance improvement of VLSI CMOS circuits by optimally sizing the transistors on the first N critical paths. The global picture of the circuit is considered by taking into account the effects that the transistor size changes of one path have on the others. The optimization technique in our sizing tool is based on simulated annealing and couples accurate delay modeling with power and area optimization. The combinatorial minimization of the objective function relies on analytical models that can accurately evaluate the delay, the power and the area of a gate. ASAP has been implemented in C on an Apollo 400 workstation with encouraging results.

1 Introduction

The synthesis of large-scale VLSI systems calls for high-performance design techniques. The merits of a high-quality design are fast switching speeds, small silicon area, and low power dissipation. It is a non-trivial task to synthesize large logic circuits that are well-optimized and meet the delay requirements. Hence, designers resort to timing verifiers[1][2][3] and timing optimizers[4][5] for tuning their designs.

To improve the switching speed of a particular circuit block on the critical path, one may seek to increase the widths of the transistors in the block, resulting in an increased current drive and hence lesser block delay and better output transition-time. It is to be noted, however, that even though the delay of this particular block is reduced, an increase in the transistor widths also increases the capacitive loading of the preceeding block (on the same path) and may adversely affect the overall circuit delay.. Moreover, an increased current drive for the present block, coupled with a slower transition-time at the output of the preceeding block (due to an increased loading), also increases the power dissipation of the circuit. Thus, the issues regarding the delay, the area, and the power dissipation are fairly interlinked. Our optimization technique attempts to size the transistors such that the resulting solution is a satisfactory tradeoff between them.

A variety of approaches has been suggested in the past for transistor sizing and performance improvement of VLSI systems. TILOS[4] has been used to size many practical circuits. It formulates the delay function as a posynomial that is unimodal and convex. For such a function, a local extremum is also a global extremum. In its search for the optimum, TILOS increases the size of the critical transistor by a fixed value during each iteration[5]. But, there can arise situations where the sizes of the transistors along a critical path need to be reduced; TILOS is not very well suited to handle such cases[7]. Any automated optimization program works in close conjunction with a timing-analysis software. The performance of the optimization scheme is initimately related to the accuracy of the timing tool. iDEAS[5] and Aesop[8] use simple linear-resistor delay models for the transistors. Ltime[9] also uses an RC-model to formulate the delay anil then solves the resulting non-linear equations by an iterative relaxation technique. The fact that the non-linear transistors in the gates are modeled by linear resistors can contribute to a source of error in the delay evaluation of the transistor circuits. In [11], the delay modeling is based on pre-characterization of the effective transition-resistances by finding proper regression coefficients. Such methods may be time and space consuming and may incorporate interpolation errors as well. Hedenstierna[10]

has used functional models for CMOS circuit speed and buffer optimization. His work is based on Shockley's square-law MOS model for the transistor operating in its saturation region. This model does not take into account the carrier-velocity saturation effect and tends to become inaccurate for short-channel devices.

In this paper we introduce a transistor sizing tool, ASAP, that minimizes the delay, the area, and the power dissipation (or a combination thereof) of a circuit by optimizing the sizes of the gates on the N most critical paths of the circuit. The critical paths are obtained from a timing analyzer. In course of the optimization process, there may arise a situation where an alteration in the size of a component can reduce the delay of a specific critical path at the expense of increasing the delay of some other path(s). In such a case, the change is effected only if the resultant maximum delay of all the paths after the size change is less than the maximum delay before the change. We take a global picture of the circuit into account and refrain from making circuit changes that actually worsen the overall performance.

The optimization in ASAP incorporates accurate analytical models in the formulation of the cost-function and uses closed-form equations for faster evaluation of the gate delay and the power dissipation. The delay and the power approximations for inverter circuits are based on Sakurai's a-power law MOSFET model[6] that accounts for the velocity saturation effect which becomes prominent in the sub-micron devices. This model has been found to be in reasonable agreement with SPICE for typical circuit sizes and loading. For static gates other than the inverter, we use a scheme very similar to the one in [12], by which each gate is mapped to an equivalent inverter having an equivalent output capacitance. The size of the inverter and the value of the modeling capacitor depend upon the gate-type, the gate size, the switching input, the input transition, and the process technology.

The optimization technique in ASAP is based on simulated annealing. Given a particular combinational logic circuit, ASAP optimizes the sizes of the transistors on the critical paths of interest in order to minimize the delay, the area, and/or the power dissipation of the circuit. It is to be noted that the closed-form expressions, that are used to predict the delay and the power of each block on a certain critical path, are functions not only of the parameters (the transistor widths) of the particular block but also of those of the preceeding and the succeeding blocks on the path. The output transition-time of the preceeeding block and the input-capacitance of the succeeding block determine, respectively, the input transition-time and the output loading of the current block. This inter-dependence of blocks makes it almost impossible to derive (and hence

optimize) a single closed form expression connecting the delay and the power dissipation of all the circuit blocks. Added to this is the need (the reason is explained in the next section) to consider a global picture of the circuit during the optimization. As a result, the objective function to be minimized takes a complicated form and simulated annealing seemed to be a prudent choice for optimizing such an objective function.

Our optimization program can be tailored to yield suitable tradeoffs between delay and area or power, depending upon which parameter is more critical for the design under consideration. This is achieved by suitably choosing the weights for these parameters. Each weight is a number in the closed interval $[0, 1]$ and determines the relative sensitivity of the optimization cost function to a change in the corresponding parameter value.

The paper is organized as follows. Some basic concepts are illustrated in Section 2. Section 3 talks about the area, the delay and the power calculations for a simple inverter. This idea is extended to the other CMOS static gates in Section 4. The details of the optimization procedure – its problem formulation and its solution technique – are given in Section 5. Section 6 presents an overall picture of how our optimization program fits into a synthesis flow. Some results and comparisons with a different optimization program are presented in Section 7. Section 8 draws the conclusion.

2 Preliminaries and Definitions

The most ***critical path*** of a circuit is that particular path along which a signal-transition takes the maximum amount of time to propagate from a primary input to a primary output. The maximum operating speed of a circuit is therefore limited by the signal propagation delay along the most critical path. When we have a target operating speed (and hence a target clock frequency) for a particular circuit, there is a maximum delay that we can tolerate in propagating a signal from the primary input(s) of the circuit to its primary output(s). In such a case, all those paths, for which the propagation delay is greater than the maximum tolerable delay, are termed as the critical paths of the circuit. The goal of our optimization program is to optimally size the transistors on the critical paths of a circuit. The program is closely tied to a critical-path analyzer that identifies a set of N critical paths. The capacitive loading of a block, which is connected not only to a next block on the critical path, but also fans out to some other non-critical-path blocks, is extracted by a parasitic extractor. The set of critical paths, along with the interconnect and the fanout loadings, from the critical and the non-critical regions of the circuit, form the input to the optimization

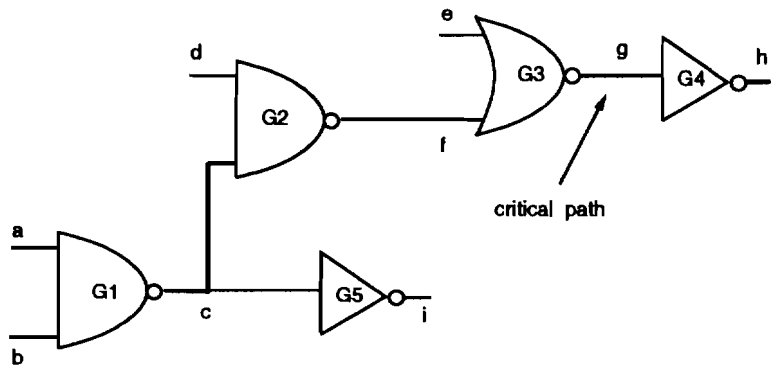

Figure 1: Capacitive loading from non-critical paths

program. The critical paths considered are allowed to have any amount of interdependencies. As an example, let us consider the circuit shown in Figure 1. b – c – f – g – h is the critical path of the circuit. The delay of the block $G1$ depends not only on the loading from the gate G2 which is the next block on the critical path, but also on the inverter G5 which is in a non-critical region of the circuit. The loading from the gate G2 is assumed to be variable, depending on the widths of the transistors constituting G2. The diffusion capacitances at the source terminals of the NMOS and the PMOS transistors of $G1$ also contribute to a variable capacitance at its output. The loading from the gate G5 and the interconnect capacitance of the output net of $G1$ are added to form a fixed capacitance. The fixed and the variable capacitances are both lumped at the output of $G1$ while modeling its delay during the optimization process.

When the transistor sizes for the critical-path logic modules are to be changed., the global effect of such a change on the delays of the other critical paths should be taken into account. In a later section we will illustrate this point with actual numerical data. For now, let us try to explain qualitatively. the necessity of such considerations. Figure 2 shows a logic circuit that has two interdependent paths: pathl (A – B – C – D) and path2 (A' – B' – C' – D). They are dependent because the NAND gate G5 is common to both of these paths. Consider this circuit to be a part of a larger block such that, for the inputs A and A' in the subcircuit, switching HIGH and LOW respectively, pathl and path2 both become critical (propagating falling and rising transitions to D). Let us assume that pathl is most critical and path2 is less critical although its delay is close to that of pathl. If we consider only one of the paths at a time (for optimization), we will consider pathl first because it is most critical. We can reduce the delay of pathl by increasing the widths of the NMOS transistors of G5 (as this gate is propagating a falling transition at its output). This increase in the transistor sizes increases the effective fan-in capacitance of G5 and it can so happen

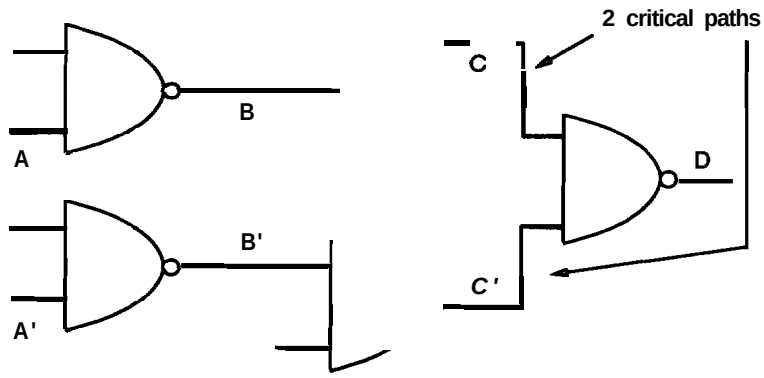

Figure 2: Consideration of multiple critical paths

that even though the delay of path1 is reduced, the increased loading capacitance, that the gate G4 now has to drive, not only makes path2 critical, but the new critical delay of path2 becomes greater than the initial delay of path1 (which we started with). Thus, even after reducing; the delay of the most criticall path, we have actually worsened the circuit performance by increasing the delay of the second critical path because it has increased the overall maximum circuit-delay. Such circuit size changes definitely need to be avoided and this can be done by considering all the critical paths at the same time (even though only one of them is being optimized). Had we done so for the circuit under consitleration, we would have readily noticed the detrimental effect (on the second critical path) of sizing only one critical path at a time. Our optimization scheme always takes the global picture into account so that it never makes a change that actually results in a drastic worsening of the overall circuit performance.

3 Modeling Inverter Delay and Power

The delay modeling of an inverter involves the calculation of not only the inverter delay but also the transition-time at the output of the inverter. Our inverter model is based on Sakurai's a-power law model[6]. The gate delay, the output transition-time, and the power dissipation depend on the input-waveform slope, the output loading and the width of the PMOS and the NMOS transistors constituting the gate. The output transition-time has to be calculated because the timing analyzer derives the output slope from the transition-time and applies it to the input of the next adjacent gate on the path of interest. We calculate the delay of a gate as the time difference between the 50% points on the output and the input waveforms. In other words, the gate delay is defined as the time difierence (t2 – t1), where $t1$ is the time corresponding to the $V_{DD}/2$ point on the input

voltage waveform, and t2 is the time when the output voltage reaches $V_{DD}/2$. The output slope is calculated from the line joining the $0.1V_{DD}$ and the $0.9V_{DD}$ points on the output voltage waveform. It is well known that a rising ramp input to an inverter discharges the output capacitance of the inverter through the NMOS transistor and for a falling input transition the output capacitance gets charged by a current through the PMOS transistor. The delay in the discharging case, t_{pHL}, or the delay in the charging case, t_{pLH}, is calculated[6] as:

$$t_{pHL}, t_{pLH} = \left(\frac{1}{2} - \frac{1 - V_T}{1 + \alpha}\right)\tau + \frac{C_L V_{DD}}{2I_{DO}}, \quad V_T = \frac{V_{TH}}{V_{DD}}$$

where C_L is the output capacitance of the inverter, incorporating both the external loading and the device-junction capacitances (these junction capacitances are technology dependent and are calculated from the relevant technology parameters), V_{TH} is the threshold voltage of the PMOS or the NMOS device depending on whether the output capacitance is charging or discharging, a is the velocity-saturation index (of the PMOS or the NMOS transistor), τ is the input transition-time, and I_{DO} is t'he drain saturation current (at $V_{GS} = V_{DS} = V_{DD}$) of the PMOS or the NMOS device. The output transition-time is given by:

$$t_T = \frac{t_{0.9} - t_{0.1}}{0.8} = \frac{C_L V_{DD}}{I_{DO}}\left(\frac{0.9}{0.8} + \frac{V_{DO}}{0.8V_{DD}} ln \frac{10V_{DO}}{eV_{DD}}\right)$$

The calculation of the short-circuit power dissipation per switching of the inverter is also based on the α-power law model and is given by:

$$P_S = V_{DD}\, \tau\, I_{DO}\, \frac{1}{\alpha + 1}\, \frac{1}{2^{\alpha-1}}\, \frac{(1 - 2V_T)^{\alpha+1}}{(1 - V_T)^{\alpha}}$$

where the symbols have their usual significance, as mentioned earlier. It is to be noted, that the terms 'input; slope' and 'input transition-time' refer to two different parameters; input slope is actually the power supply voltage divided by the input transition-time.

4 Equivalent Inverter Mapping

For the delay modeling of NAND and NOR gates, the gate is first mapped to an equivalent inverter (such that the inverter delay closely approximates the delay of the gate) and the inverter is subsequently analyzed. The equivalent inverter for each gate depends on the type of the gate (ie. the gate structure), the gate size (ie. the width of the transistors), the number of inputs the gate has, the particular input that is switching, the switching transition (ie. whether the input to the gate is rising or falling), and some technology-dependent parameters that determine the

junction capacitances associated with the transistor terminals. The inverter mapping, details of which are given in [12], consists of two parts – finding the equivalent-inverter width and calculating the modeling-capacitance at the inverter output.

The equivalent-inverter equations and the method followed to derive them are different from the ones we :proposed in an earlier paper [12]. We will not go into the details of the derivation, but will mention the results and point out the differences. Following the differences given here and the steps mentioned in [12], the derivation of the resulting formulae is straightforward. The formulae given below apply to a NAND gate; the NOR-gate equivalent can be derived by interchanging the subscripts (n and p) and the transition cases (input-rising and input-falling) for the NAND. All the PMOS transistors in a gate are assumed to be of the same size (whose widths are denoted by W_p) and the same holds true for the NMOS transitors (the widths being given by W_n) as well. For a NAND gate with N inputs, the equivalent-inverter PMOS and NMOS transistor widths, for the k^{th} gate input *(ie.* the input of the k^{th} transistor T_k) switching, are given by:

$$W_{peq} = W_p$$

$$W_{neq} = \frac{2W_n}{M+1}$$

where:

$$M = \begin{cases} N & \text{if } T_k \text{ switching HIGH} \\ k & \text{if } T_k \text{ switching LOW} \end{cases}$$

Keeping in mind that the average conductance of a switching transistor, and hence that of a switching inverter, is half the conductance of a fully turned one, the equivalent capacitance for the input-rising case can be derived as:

$$C_{eq} = \frac{(N+1)W_{neq}NC_{pd}}{2W_n} + \frac{C_{nd}W_{neq}}{2W_n}\left[2kN + 3k - N - k^2 - 1\right]$$

where C_{pd} and C_{nd} are the diffusion capacitances of the source (or the drain) of the PMOS and the NMOS transistor respectively. For the k^{th} input switching LOW, the equivalent capacitance is given by:

$$C_{eq} = \frac{W_{peq}}{W_p}[NC_{pd} + (2k-1)C_{nd}] + \frac{C_{nd}W_{peq}}{2W_n}k(k-1)$$

It is to be noted that the the equivalent-capacitance equation for the input-falling case incorporates the width of the PMOS transistor as an explicit term because it is through the PMOS transistor that the output loading capacitances of the gate, as well as the junction capacitances of the effective part of the NMOS structure, are getting charged.

Our optimization goals, as mentioned before, include both area and delay minimization. Therefore, we need mapping mechanisms for both area and delay. We derive equivalent-inverter area factors which are the ratios of the incremental gate area change to the incremental equivalent-inverter area change. The PMOS and the NMOS transistor area factors for a NAND gate are given by:

$$A_p = N$$

$$A_n = \frac{N(M+1)}{2}$$

where:

$$M = \begin{cases} N & \text{if } T_k \text{ switching HIGH} \\ k & \text{if } T_k \text{ switching LOW} \end{cases}$$

In the case of complex gates, the gate structure is first reduced to a series interconnection of transistors (with the parallel branch effects being accounted for by the inclusion of capacitances at the interconnection nodes) and then the series connection of transistors is mapped to an equivalent inverter. This algorithm, for the equivalent-inverter-mapping of complex gates, is detailed in [12].

5 Optimization by Simulated Annealing

We use simulated annealing [15] for our optimization. The choice is motivated primarily by its flexibility in terms of the forms of the objective functions that it can handle. The complexity of the objective function arises from the need to handle delay, area and power and also from the requirement of handling N critical paths simultaneously.

Any circuit can be assumed to consist of two parts (referred to as sub-circuits henceforth), critical and non-critical. Each sub-circuit comprises a set of interconnected elements. An element in our case is either a simple inverter or an equivalent inverter along with an equivalent capacitance at its output (the size of the equivalent inverter and the value of the equivalent capacitance are obtained by the modeling technique mentioned earlier). A 3-input NAND gate gives rise to three equivalent inverters (one for each input). In course of the optimization, whenever we perturb any of these equivalent inverters, we reflect this change in the sizes of the other two inverters, because a change in any of these inverters implies a resultant change in the size of the actual gate from which all these equivalent inverters have been derived. A perturbation of the width of an equivalent inverter also requires the necessary updating of the equivalent capacitance at the output

of the inverter. During the annealing process, a move from one state to another actually implies a perturbation of the transistor widths of one of the inverters in the critical sub-circuit.

If N has a high value and the critical sub-circuit has a large number of gates, the annealing method of optimization can take an enormous amount of time. An interesting observation is that, while a change in the sizes of the gates on the 'most critical path' can adversely affect the other critical paths, it is also true that the only way to reduce the maximum delay is by changing the size(s) of one or more elements on the most critical path. Therefore, although we do 'look' at all of the N critical paths in order to determine the goodness of a solution, our perturbation space is limited only to the elements of the 'most critical path'. This speeds up the annealing process immensely without affecting the quality of the result. A new path, however, might become most critical at the completion of the annealing process. We do allow this to happen because the delay of this new 'most critical path' can only be lower than the delay of the most critical path that we started with. During each annealing iteration the size(s) of the critical path components may get changed and there being the possibility of another path becoming most critical because of these changes, the current 'most critical path' is determined at the end of the iteration. The annealing process is then repeated on the new 'most critical path'. Such iterations are continued till a user-defined limit on the number of iterations or a user-defined minimum improvement ratio between successive iterations is reached.

The cost function used can be expressed as:

$$\Delta Cost = W_{delay} \times \Delta Delay + W_{area} \times \Delta Area + W_{power} \times \Delta Power$$

where,

A Cost = the incremental cost of a perturbation,

A Delay = the incremental change in the maximum delay amongst the N critical. paths,

A Area = the incremental area change, and

A Power = the incremental change in the power dissipation.

The weights W_{delay}, W_{area}, and W_{power} are provided by the user.

The cooling schedule used is the one proposed in [16].

6 Performance-driven Synthesis

The design-cycle of a chip can be largely reduced if the circuit-level optimization techniques can be efficiently applied to the design as a whole. ASAP can potentially be used in a design flow

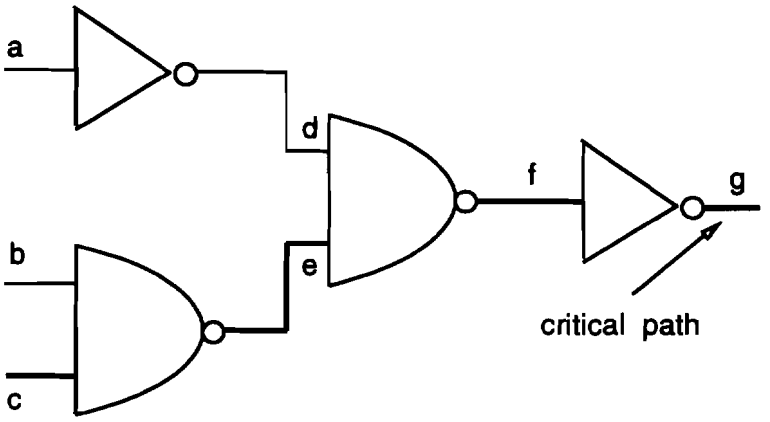

Figure 3: Single-fanout circuit example

Table 1: Delay comparison with DROID

Critical path delay (ns)		*%-age*
DROID	*ASAP*	*improvement*
0.94	0.78	20.51

for optimal circuit synthesis. There can be an iteration loop between the circuit synthesis tool, the timing analyzer, and the circuit optimizer. This loop can be executed in the pre-floorplan stage (with only the loading capacitances available), the post-floorplan and the pre-layout stage (with the interconnect-capacitance estimates) and the post-layout stage (when the exact parasitics are available). The optimization technique in ASAP takes into account the minimum and the maximum sizes of the transistors as permitted by the technology and the layout constraints, and tries to optimize the design within such bounds. It can also take as an input a target clock frequency and optimize a design to meet the timing specifications.

7 Circuit Examples and Results

Let us first consider the circuit shown in Figure 3. Table 1 compares our delay-optimization results for the above example with the results obtained from a different circuit optimization program[14]. Using a timing analyzer it is found that the path $c - e - f -$ g is most critical for a rising transition at the input c. The delay value obtained after the optimization of the circuit using ASAP is about 21 % less than that predicted by the DROID[13] synthesis system – both programs having identical limits on the objective function and the minimum and the maximum allowable transistor sizes. It is to be noted, however, that when we refer to a particular path, we actually implly two sub-paths,

Table 2: Power-delay tradeoff

Delay weight	*Power weight*	*Delay (ns)*	*Avemge Power (μW)*
1.0	0.0	0.78	1951.0
0.9	0.1	0.96	381.2
0.5	0.5	1.05	310.0
0.2	0.8	1.10	306.8

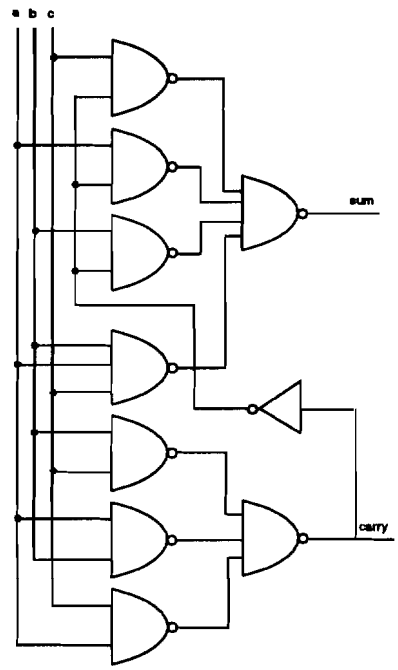

Figure 4: One-bit combinational full-adder

one for each transition (rising and falling) at the input of the path. The delay of a critical path refers to the maximum of the delays of the two sub-paths.

As mentioned earlier, the relative importance of power and/or delay optimization for a circuit can be weighted in ASAP. Table 2 shows the power-delay tradeoff for several different weight factors. For explicit illustration of the tradeoff, the area weight factor is set to zero in this case.

A full-adder circuit is shown in Figure 4. Table 3 compares our delay-optimization results for the full-adder with the results obtained from DROID. Our optimized-circuit delay is about 11 % better than that predicted by DROID.

Just like the tradeoff between power and delay, a suitable tradeoff between area and delay can also be obtained by choosing suitable weight factors. The area-delay tradeoff is illustrated in Table 4; the weightage for power optimization is set to zero in this case.

A combinational logic block with sufficiently large fanout is shown in Figure 5. Table 5 illustrates

Table 3: Delay optimization of a full-adder

Critical path delay (ns)		*%-age*
DROID	*ASAP*	*improvement*
1.89	1.70	11.18

Table 4: Area-delay tradeoff for adder optimization

Delay weight	*Area weight*	*Delay (ns)*	*Area (μm)*
1.0	0.0	1.70	143.95
0.8	0.2	1.79	98.45
0.6	0.4	1.91	51.26
0.5	0.5	1.97	45.53
0.3	0.7	2.20	63.14

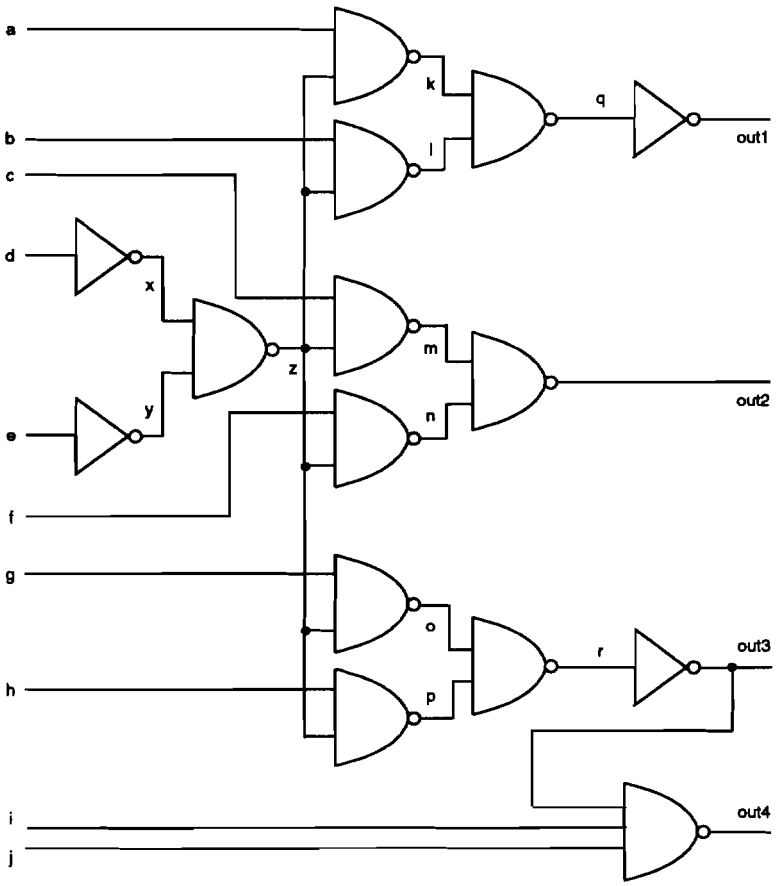

Figure 5: Combinational logic with multiple fanout

Table 5: Delay improvement of combinational logic-circuit

Critical path delay (ns)		*%-age improvement*
DROID	ASAP	
2.09	1.77	18.08

Table 6: Sizing results for the circuit in Figure 5

Critical path	*Transition at input*	*Path delay (ns)*		
		Initial setting	*Paths optimized*	
			Most critical path	*All critical paths*
e-y-z-p-r-out3-out4	R	2.085	1.694	1.773
	F	1.678	1.695	1.766
e-y-z-o-r-out3-out4	R	2.085	1.848	1.771
	F	1.678	0.876	1.773

the improvement in the circuit delay performance after the optimization using ASAP. The results are again compared with the results from DROID.

The circuit of Figure 5 can also be used to illustrate the necessity of sizing multiple critical paths at the same time. The results for single and multiple-critical-path optimizations are shown in Table 6. The initial-setting column shows the delay values obtained from a preliminary design. As is evident from the table, if we consider only one critical path for the optimization, the delay of the critical path $(e - y - z - p - r - out3 - out4)$ is reduced to 1.7 ns, but the delay of the second critical path $(e - y - z - o - r - out3 - out4)$ increases to 1.9 ns. Thus, the overall circuit-delay actually becomes worse than what we started with. This is a typical problem that arises if we neglect the effect that the transistor-size changes of one path have on the others. Such a situation is avoided if we consider multiple critical paths. In case of the current example, a consideration of multiple critical paths leads to an overall circuit delay of 1.8 ns.

8 Conclusions

In this paper, we have introduced ASAP – a simulated-annealing based transistor sizing tool that can be used for speed, power, and area optimization of static CMOS circuits. .ASAP takes into account a global picture of the circuit under consideration and sizes the transistors constituting the blocks on the first N critical paths of interest. The optimization process in. ASAP relies on functional models for accurate evaluation of the area, the delay, and the power dissipation of each

gate on the critical path(s). This program is implemented in C and has been tested on circuit examples with encouraging results.

References

[1] N. P. Jouppi, "TV: An NMOS Timing Analyzer," *Proc. Third Caltech Conference on VLSI,* pp.71-85, 1983.

[2] J. K. Ousterhout, "Crystal: A Timing Analyzer for NMOS VLSI Circuits," *Proc. Third Caltech Conference on VLSI,* pp.57-70, 1983.

[3] T. McWilliams, "Verification of Timing Constraints on Large Digital Systems," *Proc. 17th ACM/IEEE Design Automation Conference,* pp.139-147, 1980.

[4] J. Fishburn and A. Dunlop, "TILOS: A Posynomial Programming Aproach to Transistor Sizing," *Proc. IEEE International Conference on CAD,* pp.326-328, November, 1985.

[5] S. Sapatnekar and V. Rao, "iDEAS: A Delay Estimator and Transistor Sizing Tool for CMOS Circuits," *Proc. Custom Integmted Circuits Conference,* 1990.

[6] T. Sakurai and Richard Newton, "Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas," *Proc. IEEE Journal of Solid-Slate Circuits,* vol. 25, pp. 584-593, April 1990.

[7] J. Shyu, J. Fishburn, et. al., "Optimization-Based Transistor Sizing," *Proc. Custom Integmted Circuits Conference,* pp. 417-420, 1987.

[8] K. S. Hedlund, "Aesop: A Tool for Automated Transistor Sizing," *Proc. 24th ACM/IEEE Design Automation Conference,* pp.114-120, 1987.

[9] M. A. Cirit, "Transistor Sizing in CMOS Circuits," *Proc. 24th ACM/IEEE Design Automation Conference,* pp.120-124, 1987.

[10] N. Hedenstierna and K. O. Jeppson, "CMOS Circuit Speed and Buffer Optimization," *IEEE Tmns. Computer-Aided Design,* pp. 270-281, March 1987.

[11] B. Richman, J. Hansen and K. Cameron, "A Deterministic Algorithm for Automatic CMOS Transistor Sizing," *Proc. Custom Integmted Circuits Conference,* pp. 421-424, 1987.

[12] H. Chen and S. Dutta, "A Timing Model for Static CMOS Gates," ***Proc. IEEE International Conference on CAD,*** pp. 72-75, November, 1989.

[13] Stephen Lusky, Paul Kollaritsch, Doug Matzke, Derek Smith, Paul Stanford, "A Unified Design Representation Can Work," ***Proc. of the 26th ACM/IEEE Design Automation Conference,*** Las Vegas, June 1989.

[14] H. Chen, S. Agarwala, S. Dutta, et.al., "Circuit Optimization Techniques DROID," ***Symposium on VLS'I Technology, Systems, and Applications,*** pp. 162-166, May 1991.

[15] S. Kirkpatrick, C. D. Gellat, M. P. Vecchi, "Optimization by Simulated Annealing," Science, Vol. 220, N. 4598, pp. 671-680, 13 May, 1983.

[16] M. D. Huang, F. Romeo, and A. Sangiovanni-Vincentelli, "An Efficient General Cooling Schedule For Simulated Annealing," ***Proc. IEEE International Conference on CAD,*** pp. 381-384 1986.