

# A Quick Way to Find the Optimized Performance of a Power Constrained Logic Circuit

Kerry S. Lowe and P. Glenn Gulak

Department of Electrical and Computer Engineering  
University of Toronto  
Toronto, Ontario, Canada M5S 1A4

## Abstract

Single-independent path (SIP) analysis is a quick and accurate method for estimating the optimized delay of a logic circuit as a function of power constraint. The technique is based on the realization that a certain class of networks, called SIP, can be easily and precisely optimized by analytic means and that other networks can be accurately approximated by a SIP network. SIP analysis optimizes gate sizes and handles both flat and hierarchical networks. Example results confirm the utility of the technique for a diverse range of applications including network selection, data path optimization, and BiCMOS versus CMOS comparison.

## 1. Introduction

Mathematical programming based approaches for gate sizing and buffer insertion optimization can successfully find the optimized design and performance of a given combinational logic network for a specified power constraint and component library [1,2]. Such approaches, however, require significant computation and are not well suited for analysis type problems that a designer faces such as determining how optimized performance varies with changes in constraint setting and/or library data. As an alternative approach, this paper thus proposes an analysis oriented technique called *single-independent path* (SIP) analysis that efficiently and accurately estimates the optimized performance of a logic circuit.

The technique is based on the realization that a certain class of networks, called SIP, can be easily and precisely optimized by analytic means. In particular, gate sizing optimization to minimize network delay subject to a power constraint is done by simply creating the plot of constant fan-out contours as output gate size and Lagrange multiplier value is varied. For a hierarchically structured SIP network where a node represents a block of gates, block level optimization of the network is found from a closed form expression exploiting the fact that the optimized network delay-power product is equal to the square of the sum of the square root of the block delay-power products. For a general (non-SIP) network, optimized performance is estimated by deriving and optimizing a representative SIP network.

SIP analysis incorporates the results and extends the capabilities of a previous technique called *logical effort* [3]. Logical effort predicts the optimized delay of a logic circuit using the condition that after gate sizing all gates along the critical delay path have equal *stage effort* (i.e. fan-out delay) and the product of these stage efforts is equal to an optimization independent and easily calculated quantity called the *path effort*. This same result is generated, as a special case of SIP analysis, when gate sizing a SIP network to minimize delay with no power constraint.

## 2. The SIP Method

### 2.1 SIP Networks

The class of SIP networks is defined as those networks that can be mapped to a linear chain representation as per Fig. 1a. In the chain, a node corresponds to a logic component or set of equivalent components in the network while an arc corresponds to an interconnect wire or set of equivalent wires. As indicated, each of the  $D$  nodes in the chain has three parameters:  $d$  is the depth of the node from the input,  $b$  is the outdegree, and  $h$  is the number of equivalent components the node represents. Each arc in the chain has a wire capacitance parameter  $C_w$ . The output arc has a load capacitance parameter  $C_L$ . Examples of SIP networks and their chain representation are shown in Fig. 1b. Note that many types of regular array, tree, and hypercube derivative networks used in computational systems are SIP [4].

### 2.2 Analytic Optimization of SIP Networks

The goal of network optimization is to choose an implementation (from a set of available implementations) for each component such that network delay  $T$  is minimized subject to a constraint on network power  $P$  (or vice versa) and is in general a difficult task. As evident from its chain representation, however, a SIP network has just one independent delay path to be optimized. Network delay and power is then  $T = \sum \tau^d$  and  $P = \sum h^d p^d$  where  $\tau^d (p^d)$  is the implementation dependent component delay (power) of node  $d$ . Thus, by the theory of Lagrange multipliers

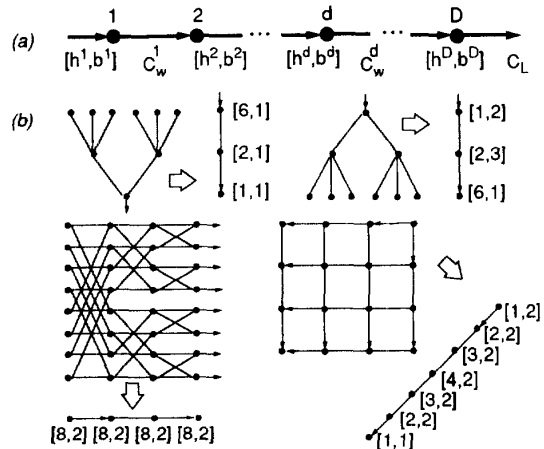


Fig. 1. SIP networks: (a) definition and (b) examples

(e.g. see [5]), minimizing  $T$  for a SIP network such that  $P \leq P_{max}$  is equivalent to minimizing the Lagrangian  $L = T + \lambda(P - P_{max})$  where  $\lambda$  is the Lagrange multiplier.

For (continuous) gate sizing optimization, different realizations for node  $d$  correspond to a different gate size  $s$  (Fig. 2a). Node (gate) delay is modeled as  $\tau = \alpha + \gamma C_{out} / (s C_{in})$  where  $\alpha$  is the internal delay,  $\gamma$  is the loaded delay factor,  $(s C_{in})$  is the input capacitance, and  $C_{out}$  is the output load capacitance or the input capacitance of the succeeding gates plus  $C_w$ . Node (gate) power is modeled as  $p = s \epsilon$  where  $\epsilon$  is a constant. Minimizing  $L$  with respect to the unknown sizes ( $s^d$  for  $d=2,3,\dots,D$ ) and  $\lambda$  then yields

$$\frac{\partial L}{\partial \lambda} = \sum_{d=1}^D h^d \epsilon^d s^d - [P_{max} - P_w] = 0, \quad (1)$$

$$\frac{\partial L}{\partial s^d} = \frac{b^{d-1} \gamma^{d-1} C_{in}^d}{s^{d-1} C_{in}^{d-1}} - \frac{b^d \gamma^d C_{out}^d}{(s^d)^2 C_{in}^d} + \lambda h^d \epsilon^d = 0$$

where  $C_{out}^d = (s^{d+1} C_{in}^{d+1} + C_w^d)$  with  $C_{out}^D = C_L$  and  $P_w$  accounts for power dissipated in the interconnect.

From (1) it is seen that  $s^d$  can be solved recursively once  $s^D$  and  $\lambda$  are known. That is,

$$\frac{1}{s^d} = \frac{C_{in}^d}{b^d \gamma^d C_{in}^{d+1}} \left[ \frac{b^{d+1} \gamma^{d+1} C_{out}^{d+1}}{(s^{d+1})^2 C_{in}^{d+1}} - \lambda h^{d+1} \epsilon^{d+1} \right] \quad (2)$$

Hence, a  $(s^D, \lambda)$  pair, where  $\lambda \geq 0$  and  $s^D > 0$ , corresponds to a solution of (1) for a particular  $P_{max}$  and  $s^1$ . The 3-dimensional plot of  $s^1$  as a function of  $s^D$  and  $\lambda$  (Fig. 2b) thus contains all solutions of (1) while the solutions of (1) for a specified  $s^1$  are contained by a contour of constant  $s^1$  (Fig. 2c). Solving (1) using (2) for points along such a contour hence generates a plot of optimum delay versus power (Fig. 2d).

In the special case of unconstrained delay minimization (with  $\{C_w\}=0$ ),  $L$  reduces to  $T$  so (2) reduces to

$$\frac{b^d \gamma^d s^{d+1} C_{in}^{d+1}}{s^d C_{in}^d} = \frac{b^{d+1} \gamma^{d+1} s^{d+2} C_{in}^{d+2}}{s^{d+1} C_{in}^{d+1}} \quad (3)$$

a result that is equivalent to the method of logical effort. Graphically, the solutions of this case correspond to the points along the  $\lambda = 0$  axes in Fig. 2b and to the (delay-power) locus indicated in Fig. 2c. As well, note that if  $D_B$  buffers are added to the path, differentiation of  $T$  with respect to  $\hat{D}_B = D + D_B$  gives the ideal number of buffers needed to achieve minimum delay [3]. Specifically,  $\hat{D}_B = \ln(F_B) / \ln \rho_B$  where path effort  $F_B$  and stage effort  $\rho_B$  are defined by

$$F_B = \frac{C_L}{s^1 C_{in}^1} \prod_{d=1}^D \frac{b^d \gamma^d}{\gamma_B} \quad \text{and} \quad \rho_B = \frac{\alpha_B}{\gamma_B (\ln \rho_B - 1)} \quad (4)$$

where  $\alpha_B$  and  $\gamma_B$  are buffer characteristics.

In a hierarchically structured SIP network where a block of gates is treated as a component (Fig. 7), the optimization task concerns the selection of a block implementation for each node  $d$ . In the case that the choice of realizations for node  $d$  arise by gate sizing a component (for a fixed  $s^1$  and  $C_L$ ), a reasonable component model is  $\tau = (\kappa/p) + \tau_0$  where  $\kappa$  and  $\tau_0$  are constants. Minimizing  $L$  with respect to the unknown component powers ( $p^d$ ) and  $\lambda$  then yields the condition

$$p^d / p^{d+1} = \sqrt{(h^{d+1} \kappa^d) / (h^d \kappa^{d+1})} \quad (5)$$

and the following closed form expression for optimized delay

$$T = \frac{\left[ \sum_d \sqrt{(h^d \kappa^d)} \right]^2}{P_{max}} + \sum_d \tau_0^d \quad (6)$$

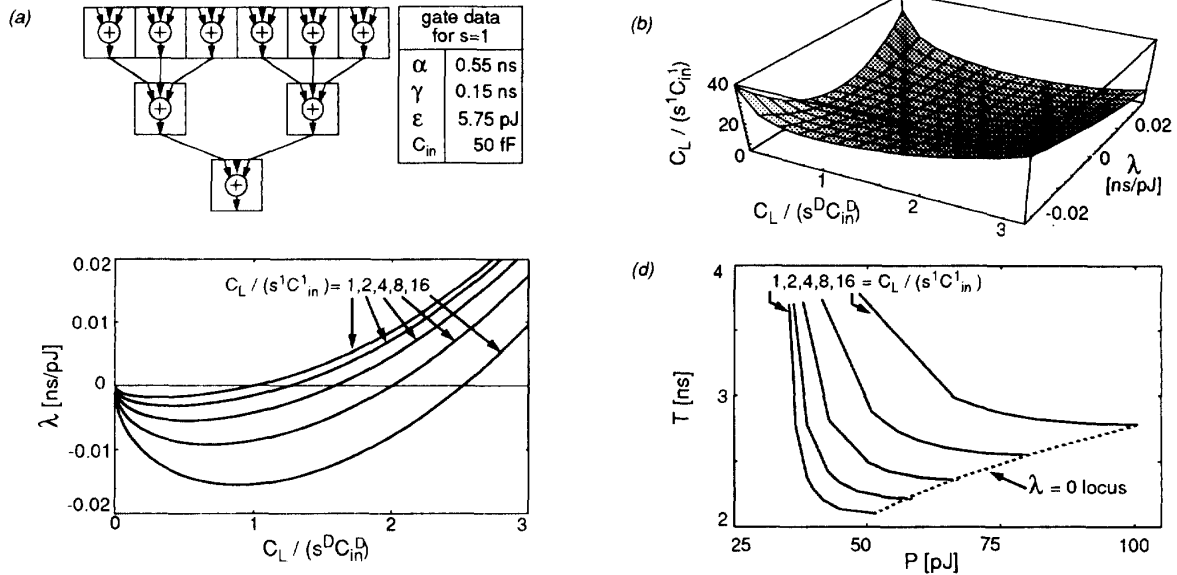


Fig. 2. SIP method for gate sizing optimization: (a) example network & library, (b) 3D-plot, (c) contour plot, (d) optimized network  $T$  &  $P$  (for  $s^1=1$ )

### 3.3.2

### 2.3 Analytic Optimization of General Networks

A general (non-SIP) network will have many delay paths to be optimized and interdependence amongst these paths will exist since a component can belong to more than one path so the methods of Section 2.2 do not strictly apply. However, it may still be possible to use an analytical approach if a SIP network can be found that approximates the general network.

Such a SIP network is derived from the delay graph of a general network (Fig. 3a) that lists the cumulative delay and delay slack at the output of each gate (when all gates are at their minimum size). Critical path(s) gates are identified as zero slack gates and a depth is assigned to each. A SIP *critical path network* is then constructed (Fig. 3b) where  $h^d$  is equal to the number of critical gates at depth  $d$  (assuming that all critical paths have the same total depth and that all critical gates at a given depth have equivalent complexity) and  $b^d$ ,  $C_w^d$  are based on an arbitrarily chosen critical path. Application of (3) to the SIP critical path network yields the maximum achievable network delay reduction  $\Delta T$  so *near-critical* gates, defined as those gates with slack less than  $\Delta T$ , are identified. Finally, the representative SIP network is derived by incrementing  $h^d$  of the SIP critical path network to account for near critical gates at depth  $d$  (Fig. 3c). This SIP network attempts to incorporate those gates in the general network that may need to be sized upward and assumes all other gates stay at minimum size.

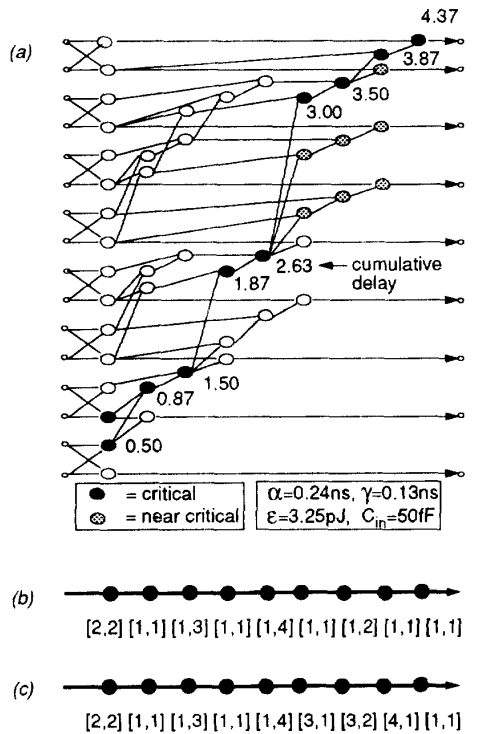


Fig. 3. Deriving SIP representation for general network (a) delay graph for cla network of Fig. 5 (with all gates at  $s=1$ ,  $C_L=100fF$ , and  $C_w=0$ ) (b) SIP critical path network (c) final SIP network

### 3. Results and Applications

#### 3.1 Buffer Chain Problem

As a first illustration of the capabilities and applications of SIP analysis, consider the classical buffer chain optimization problem of finding the number and size of buffers that minimizes delay for a given fan-out. The problem has a well known solution for the unconstrained case [6] but no closed form analytic solution in the constrained case [7]. Using SIP analysis, however, the constrained case is easily solved by applying (2) and generating the optimized delay versus power plots for different length chains (Fig. 4).

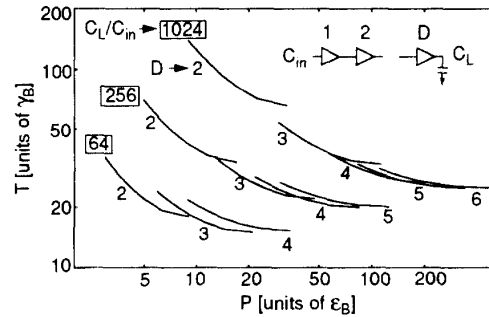


Fig. 4. Optimized buffer chain performance for different number of stages and fan-out (for case  $\alpha_B=\gamma_B$ )

#### 3.2 Network Selection Problem

SIP analysis is also useful for solving network selection problems that require a designer to choose a network from among several alternatives. Such a problem is given in Fig. 5 that shows four 8-bit adder networks following [8]. The result of applying SIP analysis to each network is shown in Fig. 6. Also shown are the results obtained by an exact optimization approach that rigorously handles effects that are only approximated in SIP analysis such as the sizing of non-critical gates and the presence of size range constraints [1]. It is evident that SIP analysis provides sufficient accuracy for networks selection purposes in the case of these general class networks.

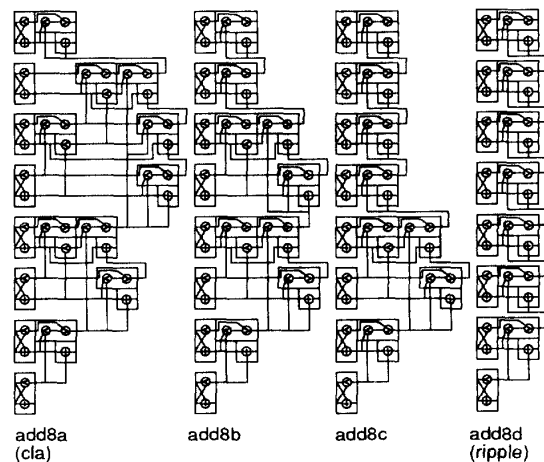


Fig. 5. Alternative 8-bit adder networks

### 3.3.3

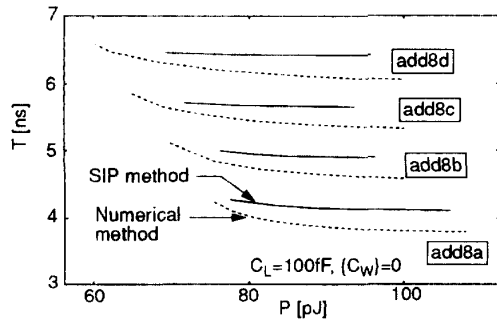


Fig. 6. Comparison of optimized 8-bit adders for gate data of Fig. 3a.

### 3.3 Data Path Optimization Problem

Data path optimization problems can also be efficiently solved by SIP analysis by exploiting its ability to handle hierarchically structured networks. Such a problem is given in Fig. 7 that shows a data path network comprising of adder and multiplier blocks that implement the DSP function of [9]. This block level SIP network is thus directly optimized using (5-6).

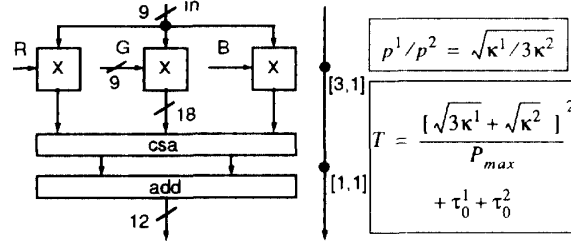


Fig. 7. Block level optimization of RGB video matrix function

### 3.4 Technology Comparison Problem

Another application for SIP analysis exists in technology comparison where a designer is given alternative libraries (i.e. different technologies) and must compare them based on the optimized performance each can deliver across a set of networks (i.e. different networks needed in a large system). For example, SIP analysis enables easy comparison of a CMOS library (containing CMOS gates) with a BiCMOS library (containing these same CMOS gates plus BiCMOS gates that can be described as a buffered form of the CMOS gates) [10]. More specifically, by using (3) and (4) for delay prediction in the CMOS and BiCMOS case, respectively, an expression is generated for the maximum speed advantage  $\mathfrak{R}$  of BiCMOS to CMOS. Namely,

$$\mathfrak{R} = \frac{DF_B^{1/D} \gamma_B + \sum \alpha^d}{\rho_B \gamma_B \hat{D}_B + \alpha_B D_B + \sum \alpha^d} \quad (7)$$

If the CMOS library itself contains buffers, (4) applies to both CMOS and BiCMOS so that the relevant expression for  $\mathfrak{R}$  is

$$\mathfrak{R}^* = \frac{\rho_C \gamma_C \hat{D}_C + \alpha_C D_C + \sum \alpha^d}{\rho_B \gamma_B \hat{D}_B + \alpha_B D_B + \sum \alpha^d} \quad (8)$$

Table 1: Performance advantage of BiCMOS versus CMOS (for  $\alpha_B=0.02$ ,  $\gamma_B=0.02$ ,  $\alpha_C=0.07$ ,  $\gamma_C=0.04$ )

Network	SIP Method	Numerical Method
	$\mathfrak{R}$ ( $\mathfrak{R}^*$ )	$\mathfrak{R}$ ( $\mathfrak{R}^*$ )
add8a ( $C_L=100fF, \{C_w\} \neq 0$ )	1.26 (1.20)	1.21 (1.17)
add8b ( " " )	1.23 (1.18)	1.18 (1.16)
add8c ( " " )	1.21 (1.18)	1.17 (1.16)
add8d ( " " )	1.19 (1.17)	1.15 (1.16)
add8a ( $C_L=400fF, \{C_w\} \neq 0$ )	1.34 (1.23)	1.27 (1.20)
add8a ( $C_L=100fF, \{C_w\}=0$ )	1.17 (1.15)	1.16 (1.14)

where a  $C$  ( $B$ ) subscript indicates a characteristic for the case of CMOS (BiCMOS) buffering. Table 1 lists the performance advantage of BiCMOS predicted using (7-8) for the networks of Fig. 5. Results obtained by rigorous gate sizing/buffer insertion optimization are also shown [1]. It is seen that SIP analysis using (7-8) is a simple and useful means for technology comparison that tracks well across changes in buffer characteristics, network topology, and parasitic loadings.

## 4. Concluding Remarks

SIP analysis gives a designer a quick way to optimize a logic circuit using simple means (e.g. hand calculator or mathematical spreadsheet). The results presented demonstrate the utility of the method for a diverse range of applications and highlight its complementary role to numerical based optimization approaches. SIP analysis is particularly suited for solving problems that arise in the initial stages of design such as choosing a network topology and technology. Once these decisions have been made, a numerical approach is useful for detailed design optimization.

## References

- [1] K. Lowe and P.G. Gulak, Gate sizing and buffer insertion for optimizing performance in power constrained BiCMOS circuits, Proc. IEEE Int. Conf. Computer-Aided Design, 1993, pp. 216-219.
- [2] D. Marple, Performance optimization of digital VLSI circuits, Tech. Report: CSL-TR-86-308, Computer Systems Laboratory, Stanford University, Oct. 1986.
- [3] I. Sutherland and R. Sproull, Logical effort: designing for speed on the back of an envelope, Proc. Advanced Research in VLSI, 1991, pp. 1-16.
- [4] F. Leighton, Introduction to parallel algorithms and architectures: arrays, trees, hypercubes, Morgan Kaufmann, 1992.
- [5] D. Luenberger, Linear and non-linear programming, Addison-Wesley, 1984.
- [6] C. Mead and L. Conway, Introduction to VLSI systems, Addison-Wesley, 1980.
- [7] S. Dhar and M. Franklin, Optimum buffer circuits for driving long uniform lines, IEEE JSSC, vol. 26, no. 1, Jan. 1991, pp. 32-40.
- [8] J. Fishburn, A depth-decreasing heuristic for combinational logic; or how to convert a ripple-carry adder into a carry-lookahead or anything in-between, Proc. DAC, 1990, pp. 361-364.
- [9] K. Lowe, J. Sitch, and M. Fortier, A 7K gate self-testable scalar-product processor for high definition video transmission application, Proc. IEEE GaAs IC Symposium, 1990, pp. 131-134.
- [10] P. Duchene and M. Declercq, Strategies for CMOS/BiCMOS gate usage on sea-of-gates arrays, Proc. IEEE CICC, 1991, pp. 14.2.1-14.2.4.