# Circuit Optimization via Adjoint Lagrangians

Andrew R. Conn, Ruud A. Haring, Chandu Visweswariah, Chai Wah Wu
IBM T. J. Watson Research Center, Yorktown Heights, NY 10598

**Abstract**

*The circuit tuning problem is best approached by means of gradient-based nonlinear optimization algorithms. For large circuits, gradient computation can be the bottleneck in the optimization procedure. Traditionally, when the number of measurements is large relative to the number of tunable parameters, the direct method[2] is used to repeatedly solve the associated sensitivity circuit to obtain all the necessary gradients. Likewise, when the parameters outnumber the measurements, the adjoint method[1] is employed to solve the adjoint circuit repeatedly for each measurement to compute the sensitivities. In this paper, we propose the adjoint Lagrangian method, which computes all the gradients necessary for augmented-Lagrangian-based optimization in a single adjoint analysis. After the nominal simulation of the circuit has been carried out, the gradients of the merit function are expressed as the gradients of a weighted sum of circuit measurements. The weights are dependent on the nominal solution and on optimizer quantities such as Lagrange multipliers. By suitably choosing the excitations of the adjoint circuit, the gradients of the merit function are computed via a single adjoint analysis, irrespective of the number of measurements and the number of parameters of the optimization. This procedure requires close integration between the nonlinear optimization software and the circuit simulation program.*

*The adjoint Lagrangian formulation has been implemented in the JiffyTune tool [12]which optimizes delay, area, slew (transition time) and power measurements by adjusting transistor widths and wire sizes. Speedups of over 35x have been realized in the gradient computation procedure by using the adjoint Lagrangian formulation, leading to speedups of up to 2.5x in the overall optimization procedure. Perhaps more importantly, these speedups have rendered feasible the tuning of large circuits. A circuit with 6,900 transistors was optimized in under two hours of CPU time.*

## 1.0 Introduction and motivation

The push towards high-performance, low-power, custom digital integrated circuits has led to a renewed emphasis on circuit optimization (tuning). This problem is best approached by means of gradient-based nonlinear optimization. In the case of dynamic tuning[10,16], function and gradient values are determined by means of a dynamic (time-domain) analysis of the underlying circuit. In the case of static optimization[11,15,17,18,19], a static timing analysis[23] is relied upon to analyze new iterates produced during the optimization process. If circuit blocks are modeled by analytic delay equations, these equations can be differentiated symbolically to determine gradients. Unfortunately, this procedure is not applicable to custom circuits because of the lack of availability of delay models for arbitrary transistor-level circuits. This problem is overcome by running dynamic simulations of each block of the design on the fly. In either type of tuning, gradient computation is often the bottleneck in the optimization procedure. Gradients are generally computed by the direct[2] or adjoint[1,3] method. The direct method requires as many simulations of the associated sensitivity circuit as the number of tunable parameters. In the adjoint method, the simulation of the adjoint circuit is repeated as many times as the number of measurements.

In this paper, we present a method by which the gradients of a circuit for the purposes of augmented-Lagrangian-based optimization can be obtained by means of a single measurement-at-a-time adjoint analysis, irrespective of the number of measurements and the number of tunable parameters. Thus the gradient computation bottleneck is ameliorated. This method of gradient computation has been applied in the context of a dynamic circuit optimization tool called JiffyTune[12]. The enhanced gradient computation allows additional constraints at a relatively low incremental cost, which is particularly significant for tuning dynamic circuits.

JiffyTune adjusts transistor widths and wire sizes to optimize power, area, delay and slews. Any combination of objective function, equality and inequality constraints is accommodated. JiffyTune uses SPECS[6,9] for fast simulation and sensitivity analysis[4,5,14]. SPECS employs piecewise approximate device models and event-driven simulation to gain an average speedup of 70x over AS/X[7,8] with a relative inaccuracy of 5%. Hence JiffyTune can tune at best to within 5% accuracy. However, simulation with AS/X before and after tuning has been used to corroborate the circuit improvements predicted by JiffyTune[12]. The function and gradient values are fed to LANCELOT, a general-purpose, large-scale nonlinear optimization package[13,21,22] which employs an augmented Lagrangian formulation and a trust-region approach.

## 2.0 Demonstration by means of an example

The concept of the adjoint Lagrangian formulation is first demonstrated using a simple example. Referring to Figure 1(a), let us assume that the circuit of interest has just one input, one output $v$, tunable parameters $x_1, x_2, \ldots, x_n$ and that the optimization problem is to minimize $d_1$, subject to $d_2 = T$, where $d_1$ is the 50% crossing point of the falling transition at the output, $d_2$ is the 50% crossing point of the rising transition and $T$ is a constant target value. The output waveform is shown in Figure 1(a). Let us assume that the nonlinear optimizer builds an augmented Lagrangian[31,32]

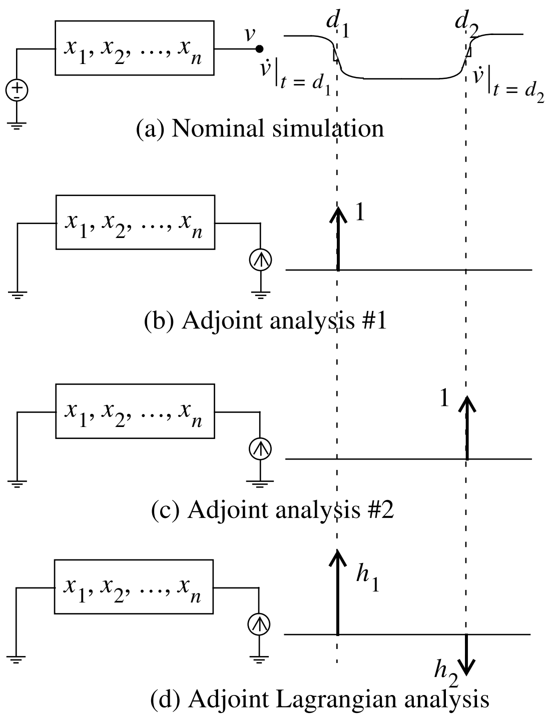



Figure 1. Demonstration of the adjoint Lagrangian formulation by means of an example.

merit function of the following form: $\Phi = d_1 + \lambda\left(d_2 - T\right) + \frac{1}{2\mu}\left(d_2 - T\right)^2$, where $\lambda$ is the Lagrange multiplier corresponding to the constraint, and $\mu$ is a penalty parameter used to weight the quadratic augmentation of the Lagrangian. At each iteration, the simulator is required to compute $d_1$, $d_2$, $\partial d_1/\partial x_i$ and $\partial d_2/\partial x_i$ for all $i$. The quantities $d_1$ and $d_2$ are computed by a nominal transient analysis (Figure 1(a)). We will first describe how the "measurement-at-a-time" adjoint analysis method is used to determine the sensitivities. First, an adjoint circuit is formed and configured as shown in Figure 1(b). The adjoint circuit is excited by a current source with a unit Dirac impulse at time $d_1$, i.e., $\delta(t - d_1)$ [34]. Time and control are reversed during the analysis of the adjoint circuit and the nominal waveforms are convolved with the adjoint waveforms to yield $\left.\frac{\partial v}{\partial x_i}\right|_{t=d_1} = \mathrm{conv}\,(x_i)$, for all $i$, where $v$ is the output and $\mathrm{conv}\,(x_i)$ represents the convolution between the appropriate nominal and adjoint waveforms for each problem variable $x_i$. The required gradients $\partial d_1/\partial x_i$ are then computed using

$$\frac{\partial d_1}{\partial x_i} = -\left.\frac{\frac{\partial v}{\partial x_i}}{\frac{\partial v}{\partial t}}\right|_{t=d_1} = -\frac{\left.\frac{\partial v}{\partial x_i}\right|_{t=d_1}}{\dot{v}|_{t=d_1}} = -\frac{\mathrm{conv}\;(x_i)}{\dot{v}|_{t=d_1}}, \quad (1)$$

where $\dot{v}|_{t=d_1}$ is the slope of the nominal voltage waveform $v$ at time $d_1$. Next, the adjoint analysis is repeated as shown in Figure 1(c) to similarly determine the gradients $\partial d_2/\partial x_i$ for all $i$. Finally, the gradients are assembled by the optimizer as follows: $\frac{\partial \Phi}{\partial x_i} = \frac{\partial d_1}{\partial x_i} + \left[\lambda + \frac{(d_2 - T)}{\mu}\right]\frac{\partial d_2}{\partial x_i}$. The method described above requires two adjoint analyses and two sets of convolution integrals. Instead, the adjoint Lagrangian method recognizes that the gradients of the merit function are the gradients of a linear combination of circuit measurements, i.e., $\frac{\partial \Phi}{\partial x_i} = \frac{\partial}{\partial x_i}\left(h_1 d_1 + h_2 d_2\right)$ where the coefficients $h_1$ and $h_2$ are known and can be treated as constants once the nominal simulation has been completed. Hence, a single adjoint analysis is enough to determine the gradients of $\Phi$ with respect to all the variables of the problem, as shown in Figure 1(c). Thus two impulses of heights $h_1 = \frac{-1}{\dot{v}|_{t=d_1}}$ and $h_2 = \frac{-1}{\dot{v}|_{t=d_2}}\left[\lambda + \frac{(d_2 - T)}{\mu}\right]$ are applied during a single adjoint analysis. Note that the heights of the impulses are functions of the nominal simulation results ($\dot{v}|_{t=d_1}$, $\dot{v}|_{t=d_2}$ and $d_2$) and the optimizer variables $\lambda$ and $\mu$. The analysis and convolution integrals are carried out as before to obtain $\frac{\partial \Phi}{\partial x_i} = \mathrm{conv}\,(x_i)$, for all $i$. Thus two adjoint analyses have been replaced by one. Although the gradients of the composite merit function can be computed by this method (see also [33,34]), the gradients of the individual measurements are not computed and cannot be recovered. In general, if an optimization problem involves $m$ measurements, the adjoint Lagrangian method will obtain a speedup of $O(m)$ over regular (one-measurement-at-a-time) adjoint analysis. The next section will describe how the gradients of any differentiable scalar merit function of any number of circuit measurements can be computed with respect to any number of parameters by means of a single adjoint analysis.

## 3.0 Theory

This section will describe how the gradients of any merit function can be computed via a single adjoint analysis. Suppose the merit function of interest is $\Phi = g(f_1, f_2, \ldots, f_F, c_1, c_2, \ldots, c_C)$ where $g$ is any differentiable function, the $f_j$ are objective functions and the $c_j$ are constraints. Further, let these objective functions and constraints be defined as differentiable functions

$$\begin{aligned} f_j &= f_j(m_1, m_2, \ldots, m_M),\ j = 1, 2, \ldots, F \\ c_j &= c_j(m_1, m_2, \ldots, m_M),\ j = 1, 2, \ldots, C \end{aligned} \quad (2)$$

of the circuit measurements $m_k$. Thus $\Phi$ can be expressed as a scalar differentiable function of the circuit measurements. We are required to find the gradient of $\Phi$ with respect to all the $x_i$, $i = 1, 2, \ldots, n$ parameters of the optimization. Now

$$\frac{\partial \Phi}{\partial x_i} = \sum_{j=1}^{F} \frac{\partial g}{\partial f_j}\frac{\partial f_j}{\partial x_i} + \sum_{j=1}^{C} \frac{\partial g}{\partial c_j}\frac{\partial c_j}{\partial x_i}, \forall i$$
$$= \sum_{j=1}^{F}\left(\frac{\partial g}{\partial f_j}\sum_{k=1}^{M}\frac{\partial f_j}{\partial m_k}\frac{\partial m_k}{\partial x_i}\right) + \sum_{j=1}^{C}\left(\frac{\partial g}{\partial c_j}\sum_{k=1}^{M}\frac{\partial c_j}{\partial m_k}\frac{\partial m_k}{\partial x_i}\right), \forall i. \tag{3}$$

Rearranging the summations,

$$\frac{\partial \Phi}{\partial x_i} = \sum_{k=1}^{M} \frac{\partial m_k}{\partial x_i}\left[\sum_{j=1}^{F} \frac{\partial g}{\partial f_j}\frac{\partial f_j}{\partial m_k} + \sum_{j=1}^{C} \frac{\partial g}{\partial c_j}\frac{\partial c_j}{\partial m_k}\right], \forall i. \tag{4}$$

All the terms inside of the square brackets of the right hand side of (4) are known once the nominal simulation has been completed, since the analytic forms of $g$, $f_j$ and $c_j$ and the nominal values of the measurements $m_k$, are known. The terms within the square brackets can depend on the nominal simulation results, as well as optimization parameters such as slack variables, Lagrange multipliers, penalty parameters, etc.,. Equation (4) can be written as $\frac{\partial \Phi}{\partial x_i} = \sum_{k=1}^{M} h_k \frac{\partial m_k}{\partial x_i} = \frac{\partial}{\partial x_i}\left[\sum_{k=1}^{M} h_k m_k\right]$, for all $i$ since the $h_k$ can be treated as constants after the nominal simulation. So far, all we have done is to write the gradients of the merit function as the gradients of a linear combination of measurements, which is always possible provided $g$, $f_j$ and $c_j$ are differentiable. Following the derivation of adjoint sensitivities in[1], we will demonstrate how to pick adjoint circuit excitations so that the gradient of the merit function can be efficiently computed.

Let each measurement be expressible as a time-domain convolution integral of the form $m_k = \int_{t_0}^{t_f} \{v_I, i_V\} p_k(\tau)\, d\tau$ where $v_I$ are voltages of independent current sources, $i_V$ are the currents of independent voltage sources, $t_0$ is the start time of the transient simulation, $t_f$ the end time, $\{v_I, i_V\}$ denotes one of $v_I$ and $i_V$ and $p_k$ is a time-domain function that will be used as the excitation in the adjoint circuit at the measurement point. Without loss of generality, all measurements can be written in terms of the voltages of independent current sources and currents of independent voltage sources, since a zero-valued current (voltage) source can always be added in parallel (series) with the voltage (current) to be measured. For example, a measurement which is a voltage value at any time $t$ is expressed as $m_k = \int_{t_0}^{t_f} v_{Ik}(\tau)\, \delta(\tau - t)\, d\tau$ so that $p_k$ is a unit Dirac impulse at time corresponding to $t$. A measurement which is a crossing time requires $p_k(\tau) = \frac{-\delta(\tau - t_{cross})}{\dot{v}|_{t = t_{cross}}}$. A power measurement which integrates the current through a voltage source from time $t_{start}$ to $t_{end}$ can be written as $p_k(\tau) = u(\tau - t_{start}) - u(\tau - t_{end})$, where $u(t)$ is the unit step function. The Elmore delay of a signal[24] can also be expressed as an integral. Expression of the measurements as convolution integrals is essential to the computation of sensitivities by the adjoint method.

Following[1], Tellegen's theorem[28] is invoked on the nominal circuit and an adjoint circuit with the same topology, but arbitrary elements. Then Tellegen's theorem is again invoked on the perturbed circuit and the adjoint circuit. The difference between the two sets of resulting equations is integrated over the time period of simulation. Time is run backwards and appropriate choices of branch constitutive relations (BCRs) are made in the adjoint circuit to yield the expression

$$\sum_I \int_{t_0}^{t_f}\left(-\delta v_I \hat{i}_I\right)dt + \sum_V \int_{t_0}^{t_f} (\delta i_V \hat{v}_V)\, dt = \sum_R \delta R \int_{t_o}^{t_f} i_R \hat{i}_R dt$$
$$+ \sum_C \left[C \hat{v}_C(t_0)\, \delta v_C(t_0) - \delta C \int_{t_0}^{t_f} \hat{v}_C \dot{v}_C dt\right] + \sum_{\substack{\text{other}\\ \text{elements}}} \ldots \tag{5}$$

In writing (5), adjoint circuit quantities are represented with a carat ( ^ ) symbol and $\delta$ is used to represent perturbations in circuit values (not to be confused with the use of $\delta$ for time-domain Dirac functions). The terms on the right hand side have been shown only for the linear resistors and capacitors in the circuit, but the equation is valid only when summed over all the elements of the circuit. Note that $\hat{i}_I$, the current of independent current sources in the adjoint circuit and $\hat{v}_V$, the voltage of independent voltage sources in the adjoint circuit are the adjoint excitations that must be chosen in order to express the sensitivity function of interest.

In general, (5) can be summarized as

$$\sum_{Is} \int_{t_0}^{t_f}\left(-\delta v_I \hat{i}_I\right)dt + \sum_{Vs} \int_{t_0}^{t_f}\left(\delta i_V \hat{v}_V\right)dt = \sum_l \beta_l \delta x_l \tag{6}$$

by proper choice of adjoint circuit elements, where $\delta x_l$ and $\beta_l$ represents the variation in the sensitivity parameters and the corresponding convolutions respectively. We manipulate the left hand side of (6) to the form $\sum_{k=1}^{M} h_k \delta m_k$ by choosing $\hat{v}_V$ and $\hat{i}_I$ to be $\hat{v}_V(\tau) = u(\tau - t_{start}) - u(\tau - t_{end})$, $\hat{i}_I(\tau) = -\delta(\tau - t_{cross})$ and $\hat{i}_I(\tau) = \frac{\delta(\tau - t_{cross})}{\dot{v}_I|_{t = t_{cross}}}$ for a power, voltage, and crossing time measurement respectively. All other sources are set to zero. These are the current and voltage excitations that would be applied *one at a time* if we were interested in finding the sensitivities of the individual measurements. Instead, we weight each these waveforms by the corresponding $h_k$. Substituting them into (6), we obtain $\delta\Phi = \sum_{k=1}^{M} h_k \delta m_k = \sum_l \beta_l \delta x_l$. As $\delta x_l \to 0$, the required sensitivities can be picked off as $\frac{\partial\Phi}{\partial x_l} = \beta_l$ in the course of a single adjoint analysis. Note that this result cannot be derived from simple superposition, since the adjoint circuit is a time-varying circuit.

# 4.0 Practical considerations

## 4.1 Features of JiffyTune

The adjoint Lagrangian formulation was implemented in the dynamic circuit optimization tool JiffyTune. Some features of JiffyTune are mentioned here. See [12] for more details.

- JiffyTune permits area, delay (crossing time) and power measurements. The objective function and weighted constraints are expressed in terms of these measurements. For example, a slew function is written as the difference between the 10% and 90% crossing times. If multiple objective functions are defined, a weighted sum of these functions is minimized. Both equality and inequality constraints are permitted. Inequalities are internally converted to equality constraints in LANCELOT by the addition of slack variables.
- Transistor widths can have simple bounds and can be ratioed to one another. Similar structures can be grouped so that in all instances of the structure, corresponding transistors have the same width, thus permitting a regular layout. Sensitivities are computed with respect to all tunable transistors and wires and then combined by chain-ruling to obtain the composite sensitivities with respect to the independent variables.
- Minimax optimization (e.g., minimizing the worst of $n$ path delays, without *a priori* knowledge of which delay is the worst) is supported by converting $\min_x \left[ \max_i \{f_i(x)\} \right]$ to $\min_{x,z} z$ subj. to $z \geq f_i(x)$, $\forall i$, where $z$ is a new variable of the problem. The variable $z$ is initialized to $\max_i \left[f_i\left(x^0\right)\right]$ after the first function evaluation, and the Lagrange multipliers corresponding to the inequalities that include $z$ are initialized to $1/n$ where $n$ is the number of minimax constraints, since the first order Kuhn-Tucker conditions[26] require that these multipliers sum up to unity at optimality.
- Each iteration of JiffyTune is expensive since it involves one SPECS simulation and, if the step is accepted, one gradient computation. To reduce the number of iterations, the optimizer takes aggressive steps wherever possible. Occasionally, such steps lead to circuit failures, wherein a signal transition of interest does not occur. Such situations are automatically detected and the trust-region radius of LANCELOT is reduced before making another attempt. If a certain number (5 is the default) of consecutive failures is detected, the optimization is abandoned. In practice, automatic recovery is almost always successful. The best results encountered during the course of the optimization are saved in all cases.
- If a circuit is tuned for optimal nominal performance, its manufacturability[20] may have been adversely affected. Therefore, two manufacturability modes are provided in JiffyTune. In the first mode, the final tuned circuit is simulated at each of the specified process corners so that the designer can obtain a quick idea of the performance spread. In the second mode, the circuit is *simultaneously* tuned at multiple process corners. Constraints are replicated at each specified process corner. An objective function for the nominal problem is converted to a corresponding minimax problem across the process corners. Similarly, a nominal minimax problem is mapped to a new minimax problem wherein the constraints of the nominal minimax problem are replicated at each process corner of interest. Thus, in all cases, the nominal problem is remapped into a new problem in a standard nonlinear optimization form, which ensures that tuning is carried out simultaneously at all the process corners.
- JiffyTune includes an easy-to-use graphical user interface in the Cadence schematic framework[29]. Designers can specify the optimization problem in a familiar "point-and-click" CAD tool environment. The system guides the designer through the specification procedure, suggesting, for example, that slew constraints be added each time an output delay measurement is added. The final tuned transistor widths as well as corresponding circuit delays can be back-annotated onto the schematic. All the tuning criteria are stored as attributes of the schematic, so that in the event of a technology re-map, new device models, new loading information or new timing requirements, the circuit can be re-tuned at the push of a button, facilitating design re-use.

## 4.2 Implementation

Some special considerations that were taken into account during the implementation of the adjoint Lagrangian formulation are listed in this section.

**Sorting of pulses and impulses.** Once the nominal simulation has been completed, the values of the measurements are known. Then the excitations for the adjoint circuit are created. The pulses and impulses of the adjoint circuit are appropriately scaled and sorted in reverse time order to be applied to the adjoint circuit in an event-driven manner. Event-driven techniques are used both to analyze the adjoint circuit and to carry out the necessary convolutions. In SPECS, the convolutions are typically between piecewise linear and piecewise constant waveforms[4,5,14]. To speed up the adjoint analysis, the time origin is shifted to the time of the first externally applied excitation to the adjoint circuit. Once all the excitations of the adjoint circuit have been applied, a faster convolution algorithm is employed since all waveforms are piecewise constant from that time onwards.

**Multiple adjoint Lagrangian groups.** SPECS allows multiple "adjoint Lagrangian groups," each of which contains a weighted sum of measurements as the sensitivity function of interest. The adjoint circuit is analyzed as many times as the number of groups, and one set of gradients is reported for each group. Multiple groups are necessary for Hessian computation (see below).

**Scaling of the adjoint circuit excitations.** Once the transient simulation has been completed in a new optimization iteration, the measurement values from SPECS are fed to JiffyTune. LANCELOT uses the measurement values to update its merit function and decides whether to accept the proposed step. If the step is rejected, gradient computation is skipped. If the step is accepted, LANCELOT provides to JiffyTune the values of the slack variables, Lagrange multipliers, penalty parameter and scale factors. It is important that these values be provided *after* any initializations, spacer steps[30] or two-step updates[27] in the optimizer, since these updates can change the optimizer variables which contribute to the merit function whose gradients are desired. JiffyTune uses these values to determine the scale factor $h_k$ to be applied for each measurement which is in the form $h_k = l_k + \sum_{i=1}^{M} n_{ki} m_i$ where $l_k$ and $n_{ki}$ are elements of a vector and sparse matrix that are created as a result of all the chain ruling described in Section 3.0.

**Hybrid scheme for the computation of the Hessian.** LANCELOT builds a quadratic model of the merit function at each iteration. The Hessian matrix is built up by low-rank quasi-Newton update methods such as the SR1 or BFGS[25] method. For a quadratic function $f(x)$ we have, for any direction $d$, $\nabla^2 f(x) d = \nabla f(x+d) - \nabla f(x)$. Analogously, the quasi-Newton condition maintains an approximate Hessian $B$ satisfying $Bd = \gamma$, where $\gamma$ is the appropriate gradient difference. The idea of a low rank update is to modify $B$ with new gradient difference information at moderate cost, while maintaining the quasi-Newton condition.

In "minor iterations" of LANCELOT, only the problem variables and slacks change. In "major" iterations terminated by sufficient stationarity, either the penalty parameter $\mu$ or the Lagrange multipliers $\lambda$ change, depending on whether sufficient feasibility has been achieved on the inner unconstrained problem. The merit function that LANCELOT builds is $\Phi = f(x) + \sum_i \lambda_i c_i(x) + \frac{1}{2\mu} \sum_i c_i^2(x)$, where $f$ is the objective function and $c_i$ are the constraints. Hence the Hessian is given by

$$\nabla^2 \Phi = \nabla^2 f + \sum_i \lambda_i \nabla^2 c_i + \frac{1}{\mu} \sum_i \left( c_i \nabla^2 c_i + \nabla c_i \nabla c_i^T \right). \quad (7)$$

At each minor iteration and at the start of each major iteration where $\lambda$ s are changed, low-rank quasi-Newton updates are used on the approximation of $\nabla^2 \Phi$. However, whenever a major iteration begins for which $\mu$ is changed, $B_{k+1} = B_k + \left( \frac{1}{\mu_{\text{new}}} - \frac{1}{\mu_{\text{old}}} \right) \sum_i \left( \nabla c_i \nabla c_i^T \right)$ is used to incrementally update the Hessian approximation. Hence, in this situation, the gradients of each constraint are required. Since each constraint may in turn depend on multiple measurements, multiple "adjoint Lagrangian groups" are used in SPECS. Thus a *hybrid scheme* is employed wherein adjoint Lagrangian gradient computation is used with just one group at every minor iteration and at the start of every major iteration where $\lambda$ s are changed, but an adjoint analysis with as many groups as the number of constraints is used at the start of each major iteration when $\mu$ has changed. Although the latter is more expensive, the reduction in iterations due to starting such major iterations with a better Hessian approximation makes up for the added computational cost. As an initial approximation at the first iteration, $\nabla^2 f$ and $\nabla^2 c_i$ are taken to be zero, and $\nabla c_i \nabla c_i^T$ is computed explicitly from the gradients of the constraints.

**Hessian updates with respect to slack variables.** By taking advantage of the form in which slack variables occur in the augmented Lagrangian, the Hessian entries with respect to slack variables can be explicitly computed. However, using explicitly computed values violates the quasi-Newton condition[25] when standard Hessian update formulas are used. Hence, we have developed modified update formulas which satisfy both the quasi-Newton condition and allow us to assign explicitly computed Hessian entries. For example, consider a problem with one inequality constraint $c$ and objective function $f$. The augmented Lagrangian function $\Phi$ is given by $\Phi = f(x) + \lambda(c(x) + s) + \frac{1}{2\mu}(c(x) + s)^2$, where $s$ is the slack variable. Clearly $\nabla_s^2 \Phi = \frac{1}{\mu}$. If this is applied after the regular Hessian updates, the quasi-Newton condition will be violated. Instead, in general, we use the rank-two update

$$B_{k+1} = B_k + \frac{1}{v_k^T E s_k} \left[ (y_k - B_k s_k) v_k^T E + E v_k (y_k - B_k s_k)^T \right] - \frac{(y_k - B_k s_k)^T s_k}{\left( v_k^T E s_k \right)^2} E v_k (E v_k)^T, \quad (8)$$

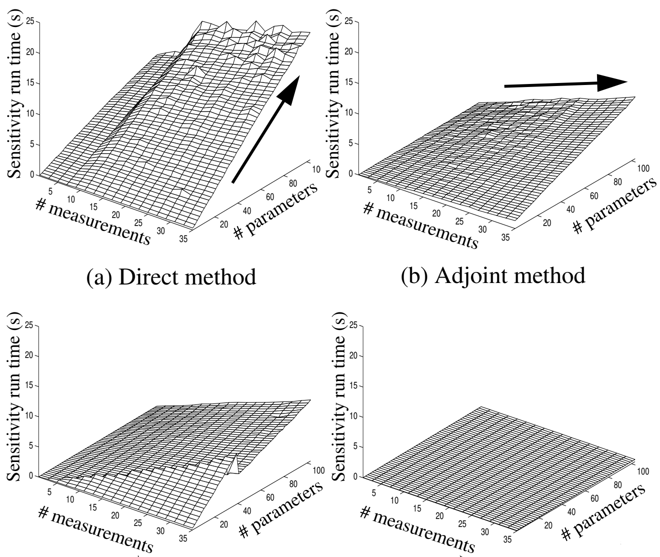
where at the $i^{th}$ iteration, $B_i$ is the Hessian approximation, $y_i$ is the change in $\nabla \Phi$, $s_i$ is the step and $E$ is a diagonal matrix with a zero on the diagonal corresponding to each slack variable and one otherwise. This update formula preserves the quasi-Newton condition[25]. If the entries of the Hessian approximation with respect to the slack variables are correctly set in $B_0$, then the update formula (8) guarantees that they remain correctly set after each update. By choosing $v_k$ as one of $(y_k - B_k s_k)$, $s_k$ and $y_k$, we obtain modified SR1 (Symmetric rank-1), PSB (Powell-symmetric-Broyden) and DFP (Davidon-Fletcher-Powell) Hessian updates, respectively. By adding the term $-\left(s_k^T E B_k s_k\right) w_k w_k^T$ to the modified DFP update where $w_k = E\left(\frac{1}{y_k^T E s_k} y_k - \frac{1}{s_k^T E B_k s_k} B_k s_k\right)$, we obtain the modified BFGS (Broyden-Fletcher-Goldfarb-Shanno) update[25]. Since the minimax auxiliary variables $z$ also appear quadratically in the augmented Lagrangian, a similar Hessian update can be derived to include them.

## 5.0 Results

JiffyTune has been used to tune a number of high-performance circuits, as reported in[12]. First, results on gradient computation are presented, followed by circuit optimization results using the new method of gradient computation.

### 5.1 Gradient computation

The adjoint Lagrangian formulation was first tested on a dynamic branch-scan-select circuit with 144 MOSFETs, an actual circuit from a high-performance PowerPC microprocessor. The number of measurements was varied from 1 to 36 and the number of tunable transistors from 1 to 104. Four analyses were conducted for each resulting combination. In the first analysis, the direct method of sensitivity computation was used. The run time of sensitivity computation as a function of the number of measurements and parameters is shown in Figure 2(a). The incremental cost of each additional sensitivity parameter is quite high (see the arrow in Figure 2(a)), as predicted by the theory. Figure 2(b) shows the run time using the adjoint method, computing the gradients of individual measurements. Again, as indicated by the arrow, the growth of run time with each additional measurement is quite high. In Figure 2(c), the run time of our previous production version is shown, in which a heuristic is used to pick the sensitivity analysis method. If the number of parameters is more than three times the number of measurements, the adjoint method is chosen. The figure shows a ridge where the program switched from the direct to the adjoint method. Finally, Figure 2(d) shows the run time of the adjoint Lagrangian formulation wherein a single adjoint analysis was used to compute the gradients of the merit function with respect to all parameters. Figure 2 clearly demonstrates not only the speedup obtained by the adjoint Lagrangian method, but also *the relatively slow growth of run time with respect to the number of parameters and number of measurements*.




(a) Direct method (b) Adjoint method

(c) Heuristic choice of method (d) Adjoint Lagrangian formulation

Figure 2. Run time of gradient computation vs. number of measurements and parameters.

The adjoint Lagrangian formulation was tested on a set of 16 benchmark circuits, whose characteristics are shown in Table 1. An adjoint sensitivity analysis was performed on each circuit to compute the individual gradients of measurements. Then the sensitivity analysis was repeated with an adjoint Lagrangian formulation, using a set of weights to form a linear combination of the measurements, as shown earlier. The gradients of the former analysis were combined in a post-processing step using the same weights, to compose the gradients of the composite merit function. The two sets of gradients were then compared. Across all the circuits, a total of 707,081 gradients were compared with the worst inaccuracy among all these gradients between regular adjoint analysis and adjoint Lagrangian analysis being 5.8e-12 (in units of either ns/μ or mW/μ ), showing that the adjoint Lagrangian formulation does indeed produce the same results.

TABLE 1. Characteristics of benchmark circuits.

| **Name** | **# MOS** | **# Ind. par.** | **# Dep. par.** | **# Meas.** | **#Constraints** | **Obj. func. ?** | **Minimax ?** |
|---|---|---|---|---|---|---|---|
| lau | 24 | 4 | 16 | 3 | 1 | Y | N |
| morrill | 8 | 3 | 3 | 2 | 1 | N | N |
| davies | 235 | 15 | 87 | 2 | 1 | Y | N |
| durham | 204 | 11 | 2 | 4 | 2 | N | N |
| Novpow | 17 | 4 | 0 | 3 | 1 | Y | N |
| fleischer | 228 | 104 | 80 | 5 | 5 | Y | Y |
| clkgen | 28 | 17 | 10 | 6 | 5 | N | N |
| Nov01 | 17 | 4 | 0 | 6 | 4 | N | N |
| nor_xor | 15 | 9 | 2 | 16 | 8 | Y | Y |
| delay | 70 | 16 | 48 | 33 | 17 | N | N |
| hot | 70 | 16 | 48 | 33 | 17 | N | N |
| cold | 70 | 16 | 48 | 33 | 17 | N | N |
| delay_minmx | 70 | 16 | 48 | 33 | 17 | Y | Y |
| hot_minmx | 70 | 16 | 48 | 33 | 17 | Y | Y |
| cold_minmx | 70 | 16 | 48 | 33 | 17 | Y | Y |
| IOmux | 6,900 | 60 | 4,068 | 82 | 41 | Y | Y |

The run times and speedups for gradient computation alone and for simulation combined with gradient computation per iteration are shown in Table 2 and graphically in Figure 3. All CPU times in this paper are on an IBM RISC System/ 6000 model 590 workstation. Unlike in Table 1, the number of measurements in Table 2 and Figure 3 excluded delay measurements on primary inputs whose gradients are known to be 0. A speedup of up to 36x is observed on circuits with a large number of measurements, which then leads to a speedup of up to 4.2x per iteration of JiffyTune. Figure 3 shows the speedup of simulation combined with gradient computation, speedup of just the gradient computation and the number of non-trivial measurements in each benchmark. From the discussion of Section 3.0, the number of measurements is an upper bound on the practically achievable speedup. On some of the smaller examples, a speedup higher than the theoretically predicted speedup is due to the granularity of CPU time measurements.

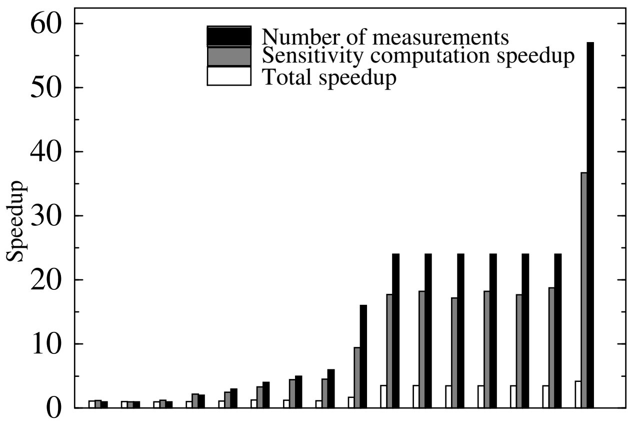


Figure 3. Histogram of total speedup, sensitivity

### 5.2 Circuit optimization

The benchmark circuits of Table 1 were optimized using the adjoint Lagrangian formulation and the new method of Hessian updates. As indicated in Section 4.2, a hybrid scheme was employed. The speedups shown in the last column of Table 2 are eroded during the optimization procedure due to an increased number of iterations and to various overheads and tasks that are common to the old and new implementations. Furthermore, because of the hybrid scheme, a "group" adjoint Lagrangian formulation is used for some major iterations. However, some gains in CPU time were observed with little loss in the quality of results. The real benefit of the adjoint Lagrangian formulation is seen in large problems, particularly problems with a large number of measurements. The results of optimizing the IOmux circuit of Tables 1 and 2 are presented in more detail below.

The IOmux circuit tuning problem was formulated as an area minimization with 41 timing constraints and a high weight on the area objective function. The area (approximated by the sum of the transistor widths) began at 31, 128 $\mu$m. The measurement-at-a-time adjoint method reduced the area to 14, 065 $\mu$m in the course of 30 optimization iterations. The total CPU time required was 270.7 minutes, consisting of 83.4 minutes of transient simulation and 187.3 CPU minutes of gradient evaluation time. With the adjoint Lagrangian formulation, after 30 iterations, the area was 15, 188 $\mu$m and the sum of the constraint violations is reduced by 20%. The run time is reduced to a total of 108.3 minutes, consisting of 83.2 minutes of transient simulation CPU time and 25.1 minutes of gradient evaluation time. Thus, the overall speedup in the optimization was 2.5x, while the speedup in the total gradient computation portion is 7.5x. Thus the gradient computation bottleneck has been effectively addressed, leaving the transient simulation as the dominant portion of the total run time!

TABLE 2. Sensitivity computation and total speedups.

| Name | # MOS | # Meas. | #Pars. | Sens.CPU time(s) Adjoint | Sens.CPU time(s) Adj. Lag. | Sens. speed-up | TotalCPU time (s) Adjoint | TotalCPU time (s) Adj. Lag. | Total speed-up |
|---|---|---|---|---|---|---|---|---|---|
| lau | 24 | 1 | 20 | 0.12 | 0.1 | 1.2 | 1.42 | 1.3 | 1.1 |
| morrill | 8 | 1 | 6 | 0.01 | 0.01 | 1.0 | 0.27 | 0.26 | 1.0 |
| davies | 235 | 1 | 102 | 0.58 | 0.48 | 1.2 | 12.4 | 12.8 | 0.97 |
| durham | 204 | 2 | 13 | 0.46 | 0.21 | 2.2 | 5.86 | 5.63 | 1.0 |
| Novpow | 17 | 3 | 4 | 0.05 | 0.02 | 2.5 | 0.47 | 0.43 | 1.1 |
| fleischer | 228 | 4 | 184 | 1.91 | 0.58 | 3.3 | 5.98 | 4.77 | 1.3 |
| clkgen | 28 | 5 | 27 | 0.31 | 0.07 | 4.4 | 1.3 | 1.05 | 1.2 |
| Nov01 | 17 | 6 | 4 | 0.09 | 0.02 | 4.5 | 0.51 | 0.44 | 1.2 |
| nor_xor | 15 | 16 | 11 | 0.66 | 0.07 | 9.4 | 1.83 | 1.07 | 1.7 |
| delay | 70 | 24 | 64 | 6.2 | 0.35 | 17.7 | 8.15 | 2.32 | 3.5 |
| hot | 70 | 24 | 64 | 6.2 | 0.34 | 18.2 | 8.14 | 2.32 | 3.5 |
| cold | 70 | 24 | 64 | 6.19 | 0.36 | 17.2 | 8.12 | 2.32 | 3.5 |
| delay _minmx | 70 | 24 | 64 | 6.2 | 0.34 | 18.2 | 8.14 | 2.33 | 3.5 |
| hot _minmx | 70 | 24 | 64 | 6.19 | 0.35 | 17.7 | 8.12 | 2.34 | 3.5 |
| cold _minmx | 70 | 24 | 64 | 6.19 | 0.33 | 18.8 | 8.13 | 2.33 | 3.5 |
| IOmux | 6,900 | 57 | 4,128 | 690 | 18.8 | 36.7 | 882 | 210 | 4.2 |

In the above example, the adjoint Lagrangian formulation reduced the CPU time of the circuit optimization from more than $4\frac{1}{2}$ hours to under 2 hours. This speedup is expected to improve further as the method is applied to larger circuits, thus rendering such optimizations feasible. Further, the adjoint Lagrangian formulation allows additional constraints at a relatively low incremental cost. This feature has a significant methodology impact, particularly for self-timed and dynamic circuits in which the number of timing "checks" that have to be satisfied during tuning can be very large.

## 6.0 Conclusions

An adjoint Lagrangian formulation for the computation of circuit gradients was proposed. For the purposes of optimization, the gradients of an augmented Lagrangian merit function can be computed in a single adjoint analysis, irrespective of the number of parameters or measurements.

Speedups of over 30x were demonstrated in the gradient computation procedure, thus addressing the bottleneck in circuit optimization programs. This gradient computation scheme has been used in JiffyTune, a dynamic circuit optimization tool, and circuits with up to 6,900 transistors have been tuned in under two hours of CPU time. The low incremental cost of additional constraints makes the optimizer amenable to tuning dynamic circuits, which typically have a large number of timing constraints. Improved methods for performing Hessian updates and better stopping criteria are currently being investigated to enhance the efficiency of the circuit optimization.

## 7.0 Acknowledgments

The authors would like to thank Ali Sadigh, Abe Elfadel and the reviewers for their suggestions on this manuscript.

## 8.0 Bibliography


[1] S. W. Director and R. A. Rohrer, "The generalized adjoint network and network sensitivities," *IEEE Trans. on Circuit Theory,* Vol. CT-16, No. 3, August 1969, pp. 318-323.

[2] D. A. Hocevar, P. Yang, T. N. Trick and B. D. Epler, "Transient sensitivity computation for MOSFET circuits," *IEEE Trans. on CAD of ICs and Systems,* Vol. CAD-4, No. 4, October 1985, pp. 609-620.

[3] R. K. Brayton and R. Spence, *Sensitivity and optimization,* Elsevier Scientific Publishing Co., Amsterdam, The Netherlands, CAD of Electronic Circuits, Vol. 21980.

[4] T. V. Nguyen, P. Feldmann, S. W. Director and R. A. Rohrer, "SPECS simulation validation with efficient transient sensitivity computation," *IEEE Int. Conf. on CAD,* November 1989, pp. 252-255.

[5] P. Feldmann, T. V. Nguyen, S. W. Director and R. A. Rohrer, "Sensitivity computation in piecewise approximate circuit simulation," *IEEE Trans. on CAD of ICs and Systems,* Vol. 10, No. 2, February 1991, pp. 171-183.

[6] C. Visweswariah and R. A. Rohrer, "Piecewise approximate circuit simulation," *IEEE Trans. on CAD of ICs and Systems,* Vol. 10, No. 7, July 1991, pp. 861-870.

[7] R. D. Kimmel, "AS/X user's guide," Tech. report, IBM GTD, Hopewell Junction, NY, August 1989.

[8] W. T. Weeks, "AS/X theory manual," Tech. report, IBM GTD, Hopewell Junction, NY, September 1989.

[9] C. Visweswariah and J. A. Wehbeh, "Incremental event-driven simulation of digital FET circuits," *Proc. 1993 Design Automation Conference,* June 1993, pp. 737-741.

[10] W. Nye, D. C. Riley, A. Sangiovanni-Vincentelli and A. L. Tits, "DELIGHT.SPICE: An optimization-based system for the design of integrated circuits," *IEEE Trans. on CAD of ICs and Systems,* Vol. CAD-7, No. 4, April 1988, pp. 501-519.

[11] J. P. Fishburn and A. E. Dunlop, "TILOS: A posynomial programming approach to transistor sizing," *IEEE Int. Conf. on CAD,* November 1985, pp. 326-328.

[12] A. R. Conn, P. K. Coulman, R. A. Haring, G. L. Morrill and C. Visweswariah, "Optimization of custom MOS circuits by transistor sizing," *IEEE Int. Conf. on CAD,* November 1996, pp. 174-180.

[13] A. R. Conn, N. I. M. Gould and Ph. L. Toint, *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A),* Springer Verlag, 1992.

[14] T. V. Nguyen, "Transient sensitivity computation and applications," Tech. report CMUCAD-91-40, Carnegie Mellon University, 1991.

[15] N. Menezes, R. Baldick and L. T. Pileggi, "A sequential quadratic programming approach to concurrent gate and wire sizing," *IEEE Int. Conf. on CAD,* November 1995, pp. 144-151.

[16] J.-M. Shyu and A. Sangiovanni-Vincentelli, "ECSTASY: a new environment for IC design optimization," *IEEE Int. Conf. on CAD,* November 1988, pp. 484-487.

[17] S. S. Sapatnekar and W. Chuang, "Power vs. delay in gate sizing: conflicting objectives?," *IEEE Int. Conf. on CAD,* November 1995, pp. 463-466.

[18] M. D. Matson and L. A. Glasser, "Macromodeling and optimization of digital MOS VLSI circuits," *IEEE Trans. on CAD of ICs and Systems,* Vol. CAD-5, No. 4, October 1986, pp. 659-678.

[19] P. K. Sancheti and S. S. Sapatnekar, "Optimal design of macrocells for low power and high speed," *IEEE Trans. on CAD of ICs and Systems,* Vol. CAD-15, No. 9, September 1996, pp. 1160-1166.

[20] R. K. Brayton, G. D. Hachtel and A. L. Sangiovanni-Vincentelli, "A survey of optimization techniques for integrated-circuit design," *Proc. of the IEEE,* Vol. 69, No. 10, October 1981, pp. 1334-1362.

[21] A. R. Conn, N. I. M. Gould and Ph. L. Toint, "Global convergence of a class of trust region algorithms for optimization with simple bounds," *SIAM J. on Numerical Analysis,* Vol. 25, 1988, pp. 433-460, See also same J., pp. 764-767, volume 26, 1989.

[22] A. R. Conn, N. I. M. Gould and Ph. L. Toint, "A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds," *SIAM J. on Numerical Analysis,* Vol. 28, No. 2, 1991, pp. 545-572.

[23] R. B. Hitchcock, Sr., G. L. Smith and D. D. Cheng, "Timing analysis of computer hardware," *IBM J. of Research and Development,* January 1982, pp. 100-105.

[24] W. C. Elmore, "The transient analysis of damped linear networks with particular regard to wideband amplifiers," *J. of Applied Physics,* Vol. 19, No. 1, 1948, pp. 55-63.

[25] P. E. Gill, W. Murray and M. H. Wright, *Practical optimization,* Academic Press, London and New York, 1981.

[26] R. Fletcher, *Practical methods of optimization,* John Wiley and Sons, Chichester, 1987.

[27] A. R. Conn, L. N. Vicente and C. Visweswariah, "Two-step algorithms for nonlinear optimization with structured applications," Res. report, IBM T.J. Watson Res. Ctr., Yorktown Heights, NY, 1997 (in preparation).

[28] B. D. H. Tellegen, "A general network theorem, with applications," *Philips Research Reports,* Vol. 7, 1952, pp. 259-269.

[29] Cadence Design Systems Inc., "Design Entry: Composer Users' Guide 4.3," 1994.

[30] D. P. Bertsekas, *Constrained Optimization and Lagrange Multipliers Methods,* Academic Press, London, 1982.

[31] M. R. Hestenes, "Multiplier and gradient methods," *J. of Optimization Theory and Applications,* Vol. 4, 1969, pp. 303-320.

[32] M. J. D. Powell, "A method for nonlinear constraints in minimization problems," in *Optimization,* R. Fletcher, ed., Academic Press, London and New York, 1969.

[33] G. D. Hachtel, R. K. Brayton, and F. G. Gustavson, "The sparse tableau approach to network analysis and design," *IEEE Trans. on Circuit Theory,* Vol. CT-18, No. 1, January 1971, pp. 101-113.

[34] R. K. Brayton and S. W. Director, "Computation of delay time sensitivities for use in time domain optimization," *IEEE Trans. on Circuits and Systems,* Vol. CAS-22, No. 12, December 1975, pp. 910-920.