High Speed Analog Circuit for Solving Optimization Problems

Kristel Deems

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-133
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-133.html

May 15, 2015

High Speed Analog Circuit for Solving Optimization Problems

By

Kristel Marie Deems

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Engineering - Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Spring 2015

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Advances in algorithms and coding have made the convergence time of a convex optimization problem reasonable for real-time optimization applications. However, solving convex optimization problems within a microsecond is still challenging, especially for larger sized problems. In addition, due to Moore's Law slowing down, the speed of modern processors is not improving at a significant rate. Many applications use parallelization in an effort to continue solving problems faster. However, this method is not effective for all convex optimization problems as it may require iterations to reach a solution, effectively increasing latency.

Fig. 1.1 shows existing real-time optimization implementations [5, 6, 8–10, 14]. Here we can see the trend of the speed of the optimizers as a function of the number of optimization variables. With the challenges now associated with improving latency of digital implementations, we propose an analog implementation as a competitive alternative, particularly for moderately sized problems as it may not be limited by the same trend in Fig. 1.1.

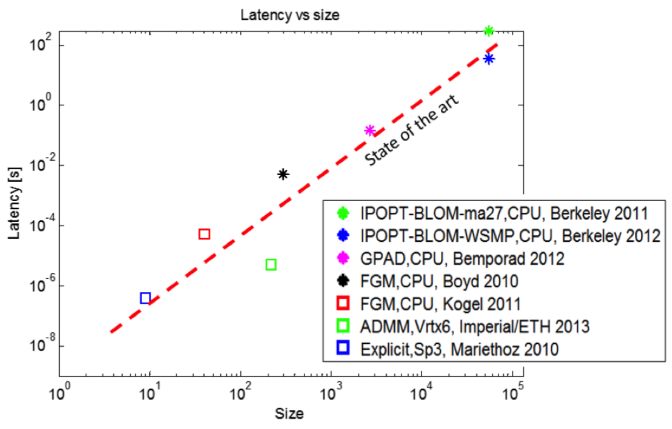

Figure 1.1: Digital optimization implementations plotted on a log scale according to latency and number of optimization variables.

Using an analog circuit to solve convex optimization problems not only improves the solution time compared to digital approaches, but could potentially have less power consumption and area as well. With the ability to solve larger problems faster, more applications become available to convex optimization, primarily real-time applications which require fast iterations. Such applications include signal and image processing, communication, and optimal control. Analog optimizers have been considered in the past and we propose a new method that can be fast and simple.

In our proposed analog circuit the steady state voltages act as the optimization solution and switched-capacitor configurations enforce the constraints. In particular, the coefficients of the constraints are set by normalized capacitance values. Using the fact that a solution to Karush–Kuhn–Tucker (KKT) conditions will in turn be the solution to the associated convex optimization problem [1], we prove that our circuit solves the desired convex optimization problem of the form:

$$\begin{aligned} &min \frac{1}{2} V^T Q V \\ &st. A_{eq} V = b_{eq} \\ &\quad A_{ineq} V \leq b_{ineq} \end{aligned} \tag{1.1}$$

V is a vector of optimization variables, A_{eq} and A_{ineq} are matrices of constraint coefficients, Q is a matrix definining the coefficients of the cost function, and b_{eq} and b_{ineq} are vectors.

1.2 Previous Work on Analog Optimization

Solving quadratic problems (QP) and linear problems (LP) in real-time with an analog circuit was first proposed by Dennis in 1959 [3]. His circuit consisted of voltage sources, current sources, diodes, transformers, and resistors. In this implementation, both voltages and currents act as optimization variables and the coefficients of equality and inequality constraints are set by the number of wires connected to a single node. This limits the coefficient values to small integers and reduces the range of solvable problems.

Chua [2] proposed a different analog circuit to solve non-linear optimization problems which was later expanded by both Chua [7] and Hopfield [12]. A non-linear optimization problem does not include equality constraints and thus has the form:

$$\begin{aligned} &\min_x f(x) \\ &\text{s.t. } g_j(x) \leq 0, \ j = 1 \ldots m \end{aligned} \tag{1.2}$$

These circuits directly implement KKT conditions in order to solve an optimization problem. Rather than enforcing the cost function, Chua and Hopfield use the Lagrangian of the optimization problem. Diodes and nonlinear function building blocks [11] are used to implement the function in (1.3).

$$\frac{\partial x_i}{\partial t} = -\left[\frac{\partial f(x)}{\partial x_i} + \sum_{j=1}^{m} I_j \frac{\partial g_j(x)}{\partial x_i}\right], \tag{1.3}$$

Here, $\frac{\partial x_i}{\partial t}$ is the current charging a capacitor. Therefore, when the circuit reaches a steady state, the capacitor charge is constant ($\frac{\partial x_i}{\partial t} = 0$) and the equation (1.3) becomes the derivative of the Lagrangian of the intended optimization problem. However, from the results of a PCB implementation [7], equilibrium is met in tens of milliseconds with a $\pm 2.5\%$ error in the solution.

This work is an extension of the work done by Vichik [13] in which the steady state voltages are variables and their weights are set by resistors. Vichik implemented the same simple problem performed by Chua and Hopfield using this method and reached the solution with a convergence time of 6 μs.

Chapter 2

Proposed Analog Optimization Circuit

2.1 Equality Constraint

For our proposed circuit, the optimization variables (V) will be node voltages and the coefficients of the constraints (A_{eq} and A_{ineq}) will be set by capacitance ratios. Consider Fig. 2.1, in which a simple equality constraint $C_1V_1 + C_2V_2 = 0$ is implemented. To more directly compare this result with the format of an optimization problem as introduced before, the equation can be rewritten as $\frac{C_2}{C_1}V_2 = -V_1$. Then $A_{eq} = \frac{C_2}{C_1}$, $V = V_2$, and $b_{eq} = -V_1$.

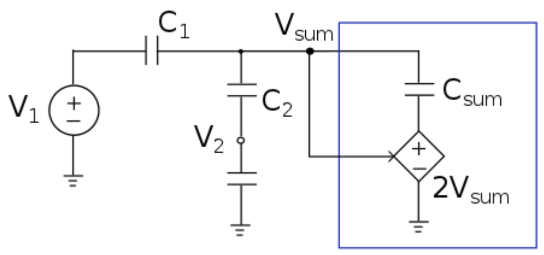

Figure 2.1: Circuit diagram demonstrating the concept behind implementing an equality constraint. This particular circuit implements $C_1V_1 + C_2V_2 = 0$.

The equation which the voltages of this circuit satisfy is determined by charge distribution at the summing node V_{sum}. Assuming that there are switches configured such that the capacitors are shorted in the first phase and the second phase is as seen in Fig. 2.1, the value of V_2 will be a function of V_1 and V_{sum} as shown in the equation (2.1). To eliminate this dependency on V_{sum}, we add a block that applies a negative voltage across C_{sum}. We see that if we're applying $-V_{sum}$ across C_{sum}, C_{sum} must be equal to $C_1 + C_2$ for our desired equation to hold.

$$\begin{aligned}
&C_1(V_{sum} - V_1) + C_2(V_{sum} - V_2) + C_{sum}(V_{sum} - 2V_{sum}) = 0 \qquad (2.1)\\
&C_1V_1 + C_2V_2 = (C_1 + C_2 - C_{sum})V_{sum}\\
&C_1V_1 + C_2V_2 = 0
\end{aligned}$$

In the general case for an equality constraint there can be multiple optimization variables, each with their own capacitive weight connecting them to the summing node. This concept is shown in Fig. 2.2. These voltages can be externally applied, thus contributing to the b_{eq} term, or they can have load capacitances that determine the voltage value the node settles to, making the node voltage a variable. In either case, the equation derived from charge distribution at the summing node will enforce the equality constraint when $C_{sum} = \sum_{i=1}^{n} C_i$.

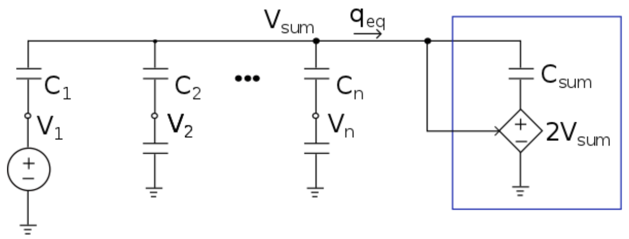

Figure 2.2: General equality constraint implementing $\sum_{i=1}^{n} C_iV_i = 0$.

$$\begin{aligned}
&\sum_{i=1}^{n} C_i(V_{sum} - V_i) + C_{sum}(V_{sum} - 2V_{sum}) = 0 \qquad (2.2)\\
&(\sum_{i=1}^{n} C_i - C_{sum})V_{sum} - \sum_{i=1}^{n} C_iV_i = 0\\
&\sum_{i=1}^{n} C_iV_i = 0
\end{aligned}$$

Note that the optimization variables $V_2, ..., V_n$ still have the freedom to change values depending on the capacitors connected at the voltage node, but the circuit has been configured such that the equality constraint is still enforced. As shown in Section 2.5, the load capacitances are determined from the desired cost function as they influence the cost function the circuit implements. Otherwise, we can choose the load capacitances to maximize the dynamic range of the variables. Smaller load capacitors decrease the value of V_{sum} and improve the dynamic range of the variables given a maximum swing constraint for V_{sum}.

2.2 Inequality Constraint

The circuit for a single inequality constraint is similar to that of an equality constraint, but with an added diode at the summing node as seen in Fig. 2.3. When $V_{sum} > V'_{sum}$, the diode is closed and the circuit behaves as an equality constraint with $V_{sum} = V'_{sum}$. With the diode closed we can rewrite the function the circuit enforces in the form

$$\sum_{i=1}^{n} C_i V_i = q_{ineq} + V_{sum} \sum_{i=1}^{n} C_i = q_{ineq} + V'_{sum} \sum_{i=1}^{n} C_i = 0 \tag{2.3}$$

$$q_{ineq} = -V'_{sum} C_{sum} \tag{2.4}$$

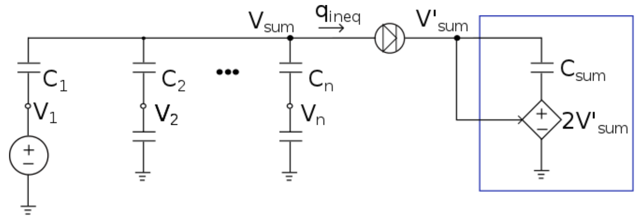

Figure 2.3: General inequality constraint implementing $\sum_{i=1}^{n} C_i V_i \leq 0$.

When $V_{sum} < V'_{sum}$, the diode is open and the optimization variables will be set by other subcircuits. In this case, the node voltages will have some dependency on V_{sum}, but they must still satisfy any other constraint circuits they are connected to. In this way, the diode always enforces $V_{sum} \leq V'_{sum}$. Using this inequality, we can rewrite (2.3) as (2.5). Then the circuit implements an inequality constraint which is only a function of the variable nodes and capacitances.

$$\sum_{i=1}^{n} C_i V_i = q_{ineq} + V_{sum} \sum_{i=1}^{n} C_i \leq q_{ineq} + V'_{sum} \sum_{i=1}^{n} C_i = 0 \tag{2.5}$$

$$\sum_{i=1}^{n} C_i V_i \leq 0 \tag{2.6}$$

The diode itself should enforce the equations (2.7). These will be used in Section 2.4 to characterize the circuit.

$$q_{ineq} \geq 0 \tag{2.7a}$$

$$q_{ineq}(V_{sum} - V'_{sum}) = 0 \tag{2.7b}$$

2.3 Implementing Multiple Constraints

For a problem with multiple equality and inequality constraints the constraint blocks are connected at the variable nodes as seen in Fig. 2.4. The figure only shows equality constraints but inequality constraints would be connected in the same way. We can see that there are m summing nodes, meaning m different constraints are enforced on n variables. Refering back to the form of a convex optimization problem, we are implementing the constraint matrix A

where $A_{ij} = C_{ij}$. Therefore, A will have m rows and n columns, where each row consists of the capacitors from a single constraint. Note that any variable can be driven by a voltage source and every constraint does not need to include all n variables. For example, if $A_{11} = 0$ then $C_{11} = 0$ and V_1 is not connected to the first constraint block.

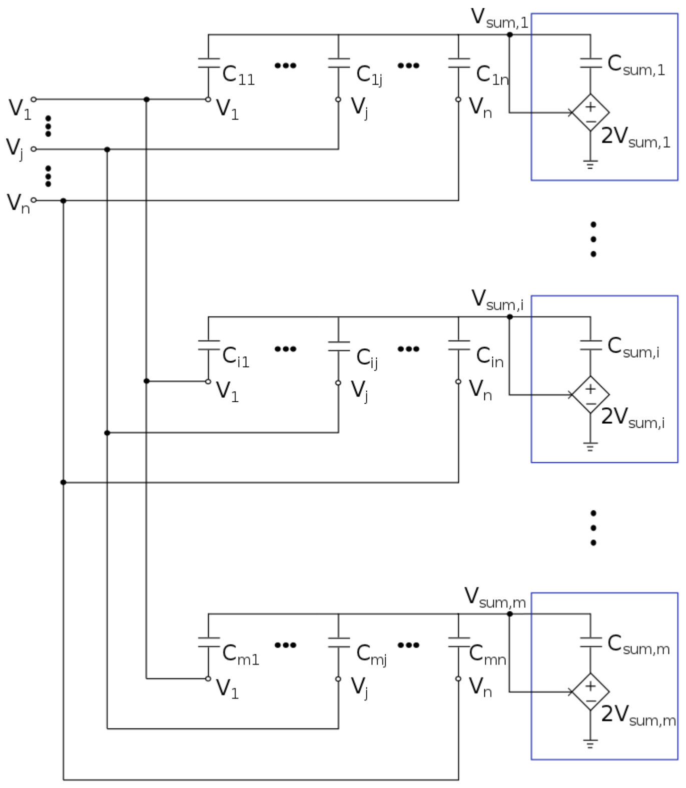

Figure 2.4: General optimization problem which implements m equality constraints with n total variables. A capacitor C_{ij} acts as a weight for the variable V_i in the jth constraint block.

2.4 Cost Function

There is no specific subcircuit that implements the cost function. Rather, the final circuit with multiple equality and inequality constraints will inherently solve a cost function which is controlled by introducing additional constraints. We can determine the implemented cost function using KKT conditions and equations from our circuit. As mentioned previously, every optimization problem can be rewritten in the form of KKT conditions. Then the solution to the KKT conditions is also the solution of the optimization problem. In this section, we show that a circuit of equality and inequality constraints satisfies KKT conditions for optimization, and therefore solves a cost function which we derive to be (2.8) [13]. Note that V is a vector that includes both constants and variables. The constants are simply the variable nodes in the circuit which are driven by voltage sources. Therefore, the cost function can include both quadratic and linear terms.

$$\begin{aligned} & min\ \frac{1}{2}V^TQ_AV \\ & Q_A = diag(1^TA) - A^Tdiag(1^TA^T)^{-1}A \\ & A = \begin{bmatrix} A_{eq} \\ A_{ineq} \end{bmatrix} \end{aligned} \tag{2.8}$$

For the following derivations, let $U = V_{sum}$, where V_{sum} is a vector of the V_{sum}'s for each equality and inequality constraint. Then we can characterize the circuit with the equations (2.9). The equations (2.9a) and (2.9b) are derived from the charge at each summing node. Equation (2.9c) is derived from the charge at each variable node. The equations (2.9d) are the imlemented equality and inequality constraints, and the equations (2.9e) are enforced by the diodes in the iequality circuits. These derivations can be seen in more detail in Vichik's work [13].

$$A_{\text{eq}}V = \text{diag}\left(\mathbf{1}^TA_{\text{eq}}^T\right)U_{\text{eq}} + q_{\text{eq}} \tag{2.9a}$$

$$A_{\text{ineq}}V = \text{diag}\left(\mathbf{1}^TA_{\text{ineq}}^T\right)U_{\text{ineq}} + q_{\text{ineq}} \tag{2.9b}$$

$$A_{\text{eq}}^TU_{\text{eq}} + A_{\text{ineq}}^TU_{\text{ineq}} = \text{diag}\left(\mathbf{1}^TA\right)V \tag{2.9c}$$

$$A_{\text{eq}}V = b_{\text{eq}},\ A_{\text{ineq}}V \le b_{\text{ineq}} \tag{2.9d}$$

$$[A_{\text{ineq}}V - b_{\text{ineq}}]_i\,[q_{\text{ineq}}]_i = 0, \forall i \in \mathcal{I},\ q_{\text{ineq}} \ge 0 \tag{2.9e}$$

The KKT conditions [1] we need to satisfy are (2.10). The free variables from these conditions are $V^\star$, $\mu^\star$, and $\lambda^\star$. Likewise, Q is an unknown constant and we can solve for its value such that the conditions are satisfied.

$$A_{\text{eq}}^T\mu^\star + A_{\text{ineq}}^T\lambda^\star + QV^\star = 0 \tag{2.10a}$$

$$A_{\text{eq}}V^\star = b_{\text{eq}} \tag{2.10b}$$

$$A_{\text{ineq}}V^\star \le b_{\text{ineq}} \tag{2.10c}$$

$$\lambda^\star \ge 0 \tag{2.10d}$$

$$(A_{\text{ineq}}V^\star - b_{\text{ineq}})_i\lambda_i^\star = 0,\ i \in \mathcal{I}, \tag{2.10e}$$

We choose Q, $U_{\text{eq}}^\star$, $U_{\text{ineq}}^\star$, $q_{\text{eq}}^\star$ and $q_{\text{ineq}}^\star$ as described by the following equations. Note that these are functions of the free variables mentioned before. Then equations (2.11) are combined with (2.10) to obtain (2.12), which are of the same form as (2.9).

$$\begin{aligned} Q =& \text{diag}\left(\mathbf{1}^TA\right) - A_{\text{eq}}^T\,\text{diag}\left(\mathbf{1}^TA_{\text{eq}}^T\right)^{-1}A_{\text{eq}} \\ & - A_{\text{ineq}}^T\,\text{diag}\left(\mathbf{1}^TA_{\text{ineq}}^T\right)^{-1}A_{\text{ineq}} \end{aligned} \tag{2.11a}$$

$$q_{\text{eq}}^\star = \text{diag}\left(\mathbf{1}^TA_{\text{eq}}^T\right)\mu^\star \tag{2.11b}$$

$$U_{\text{eq}}^\star = \text{diag}\left(\mathbf{1}^TA_{\text{eq}}^T\right)^{-1}A_{\text{eq}}V^\star - \mu^\star \tag{2.11c}$$

$$q_{\text{ineq}}^\star = \text{diag}\left(\mathbf{1}^TA_{\text{ineq}}^T\right)\lambda^\star \tag{2.11d}$$

$$U_{\text{ineq}}^\star = \text{diag}\left(\mathbf{1}^TA_{\text{ineq}}^T\right)^{-1}A_{\text{ineq}}V^\star - \lambda^\star. \tag{2.11e}$$

In particular, substitution of (2.11b) into (2.11c) and of (2.11d) into (2.11e) yields equations (2.12a) and (2.12b) respectively; substitution of (2.11a), (2.11b) and (2.11d) into (2.10a) yields (2.12c); substitution of (2.11d) into (2.10d) and into (2.10e) yields (2.12f) and (2.12g) respectively.

$$
\begin{aligned}
&A_{\text{eq}} V^\star = \operatorname{diag}\left(\mathbf{1}^T A_{\text{eq}}^T\right) U_{\text{eq}}^\star + q_{\text{eq}}^\star && \text{(2.12a)}\\
&A_{\text{ineq}} V^\star = \operatorname{diag}\left(\mathbf{1}^T A_{\text{ineq}}^T\right) U_{\text{ineq}}^\star + q_{\text{ineq}}^\star && \text{(2.12b)}\\
&A_{\text{eq}}^T U_{\text{eq}}^\star + A_{\text{ineq}}^T U_{\text{ineq}}^\star = \operatorname{diag}\left(\mathbf{1}^T A\right) V^\star && \text{(2.12c)}\\
&A_{\text{eq}} V^\star = b_{\text{eq}} && \text{(2.12d)}\\
&A_{\text{ineq}} V^\star \leq b_{\text{ineq}} && \text{(2.12e)}\\
&q_{\text{ineq}}^\star \geq 0 && \text{(2.12f)}\\
&(A_{\text{ineq}} V^\star - b_{\text{ineq}})_i {q_{\text{ineq}}}_i^\star = 0,\ i \in \mathcal{I}. && \text{(2.12g)}
\end{aligned}
$$

In conclusion, we have shown that the equations (2.12) satisfy KKT conditions and are equivalent to equations (2.9) for the Q defined in (2.11a). Therefore, the circuit enforces a cost function defined by the matrix Q which is derived from the given set of constraint matrices A_{eq} and A_{ineq} from the circuit.

2.5 Redundant Constraints

In order to implement the desired cost function without changing the equality and inequality constraints the circuit enforces, we add redundant constraints. Redundant constraints can either be equality constraints that are linearly dependent on existing equalities or they can be inequality constraints that are less than or equal to infinity. Say we're implementing a single constraint $x + 2y = 100$. Then (2.13) shows some examples of redundant constraints.

$$
\begin{aligned}
\bar{x} + 2\bar{y} &= -100 && \text{(2.13a)}\\
x + \bar{x} &= 0 && \text{(2.13b)}\\
x &\leq \infty && \text{(2.13c)}\\
x + y &\leq \infty && \text{(2.13d)}
\end{aligned}
$$

Because the equation (2.13d) is enforcing less than infinity, the inequality constraint diode is always open. As such, we can represent this constraint as a capacitance between two variable nodes as shown in Fig. 2.5. This is essentially two capacitors in series where the middle node would be the summing node. On the other hand, the redundant equality constraints such as (2.13a) and (2.13b) each require their own amplifier, so in the interest of saving power and area it's better for us to use redundant inequality constraints. The example (2.13c) is implemented in the same way as an inequality between two variables, but this time the second variable is ground.

To determine what capacitors to add given a desired cost function, we use a MATLAB-based convex optimizer called CVX [4]. The code is shown below. The format of the function

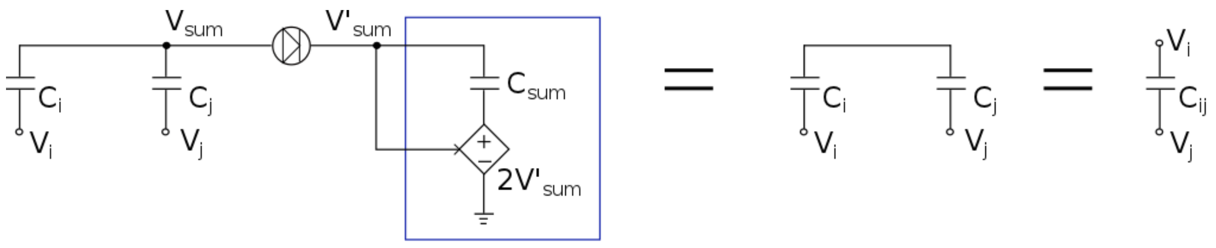

Figure 2.5: Transformation of redundant inequality between two variables

is to first list the optimization variables, then the cost function followed by the constraints the cost function is subject to. Looking at the code, H is the desired cost function matrix and must be positive definite and strictly diagonally dominant for this problem to be feasible. Q_A is the inherent cost function matrix and $M = [I \quad -I]^T$ is the transformation matrix from x and $\bar{x}$ such that $x = M[x^T \ \bar{x}^T]^T$. The matrix $Alph$ is a nxn matrix that represents all possible redundant constraint capacitors and dQ represents the effect of these possible redundant constraints on the cost matrix.

```
M = [eye(n/2) ; -eye(n/2) ];
cvx_begin
    variable k
    variable Alph(n,n) symmetric
    variable dQ(n,n) symmetric
    expression DeltaQ(n,n)
    minimize (norm(Alph(:),1))
    subject to:
        k >= 0 ;
        Alph >= 0;

        for i=1:n
            for j=i:n
                DeltaQ = DeltaQ + DeltaQmat(i,j,n)*Alph(i,j);
            end
        end
        DeltaQ == dQ;
        k*M' * H * M == M' * (Q_A + dQ) * M;
cvx_end
```

The MATLAB optimizer will minimize the amount of additional capacitance required to implement cost matrix H. This minimization occurs subject to the constraint which equates the resulting cost function matrix $Q\_A + dQ$ with a positive factor of the desired cost function matrix H. Then the solution dQ contains the normalized capacitor values needed to implement the cost function. The optimization problem this code implements is shown below.

$$\min_{\alpha_{i,j},\ 1<i,j<n} \sum_{i=1}^{n}\sum_{j=1}^{n} |\alpha_{i,j}| \tag{2.14}$$

$$s.t.\ kM^T HM = M^T(Q_A + \sum_{i=1}^{n}\sum_{j=1}^{n} \Delta Q_{i,j}\alpha_{i,j})M$$

$$k \geq 0$$

$$\alpha_{i,j} \geq 0,\ 1 < i,j < n$$

A single capacitor added between two optimization variable nodes V_i and V_j has a normalized value of α_{ij} and will implement the redundant inequality constraint $2\alpha_{ij}V_i + 2\alpha_{ij}V_j \leq \infty$. Solving for the new cost matrix, the redundant inequality constraint adds two diagonal terms of value α_{ij} and two off diagonal terms of value $-\alpha_{ij}$. The locations of these terms are shown in the DeltaQmat function below.

```
function D = DeltaQmat(i,j,n)

D = zeros(n);
if (i~=j)
    D(i,j) = -1;
    D(j,i) = -1;
    D(i,i) = 1;
    D(j,j) = 1;
end
```

2.6 Constructing the Proposed Circuit

To summarize, we can implement the general optimization problem (2.15), where the vector V represents the n variable nodes of the circuit, some of which can be driven by voltage sources.

$$min\ \frac{1}{2}V^T QV \tag{2.15}$$

$$st.\ A_{eq}V = 0$$

$$A_{ineq}V \leq 0$$

To construct the proposed circuit, we would first create all the equality and inequality subcircuits using the coefficients from the constraint matrices A_{eq} and A_{ineq} to determine the capacitor values. First we choose a unit capacitance which all other capacitors will be normalized against. Then these normalized values should be equal to the coefficients in A_{eq} and A_{ineq}. From here, we know all the capacitor values to implement our constraints. The equality and inequality subcircuits are then connected at the voltage nodes as described in Section 2.3.

The next step is to calculate the cost function $Q_A = diag(1^T A) - A^T diag(1^T A^T)^{-1} A$ which is implemented by these subcircuits. The calculated cost matrix Q_A and the desired matrix Q are called from the MATLAB function introduced in Section 2.5, which will output an nxn matrix of normalized capacitor values. These values indicate the capacitors to place between variable nodes in order to implement the required cost matrix Q. After the redundant constraint capacitors are inserted into the circuit, the proposed circuit is complete and will solve the optimization problem (2.15).

Chapter 3

Implementation Considerations

3.1 Fully Differential Equality Constraint

We chose to implement a fully differential circuit due to advantages such as high output swing and reduced noise. We use a fully differential amplifier to effectively apply a negative voltage across C_{f2} as seen in Fig. 3.1. Charge flow from the positive feedback enforces the function $V_O^+ = -\frac{C_{f1}}{C_{f2}} V_{sum}^+$. Then the negative feedback applies a negative voltage of $V_{sum}(1 - \frac{C_{f1}}{C_{f2}})$ across C_{f2} resulting in the charge analysis equation (3.1).

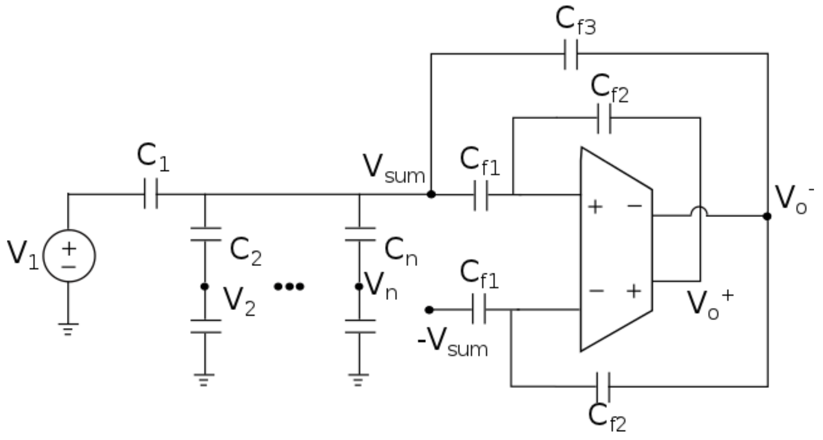

Figure 3.1: Fully differential equality constraint. For simplicity, only half of the circuit is shown.

$$\sum_{i=1}^{n} C_i(V_{sum} - V_i) + C_{f1}V_{sum} + C_{f3}(V_{sum} - \frac{C_{f1}}{C_{f2}}V_{sum}) = 0$$
$$\sum_{i=1}^{n} C_i + C_{f1} + C_{f3}(1 - \frac{C_{f1}}{C_{f2}})V_{sum} - \sum_{i=1}^{n} C_iV_i = 0 \quad (3.1)$$

There is freedom in choosing the feedback capacitors as long as they satisfy the equation (3.2) which insures the equality constraint is implemented. The ratio between C_{f1} and C_{f2} determines the amount of voltage applied across C_{f3}. A smaller ratio would decrease the output and increase the dynamic range of the system but also requires a larger C_{f3} capacitance. For simplicity, we chose a ratio of 2 and $C_{f3} = 2C_{sum}$.

$$\sum_{i=1}^{n} C_i + C_{f1} + C_{f3} = \frac{C_{f1}}{C_{f2}} C_{f3} \tag{3.2}$$

$$\sum_{i=1}^{n} C_i + C_{sum} + 2C_{sum} = 4C_{sum}$$

$$C_{sum} = \sum_{i=1}^{n} C_i$$

$$\sum_{i=1}^{n} C_i V_i = 0$$

With these choices, the complete fully differential equality constraint is shown in Fig. 3.2. We can also view this circuit as implementing two seperate equality constraints with the added condition that $V_j^+ = -V_j^-$ for all variables. This gives the same result as observing the implementation differentially with voltages $V_j = V_j^+ - V_j^-$. Therefore, the fully differential implementation has the advantage of inherently including the complement of each variable without requiring additional circuitry. This is useful when adding redundant constraints to the circuit. For example, adding an inequality constraint between x and $\bar{x}$ is an option and has the same effect on the cost function as adding capacitors from x and $\bar{x}$ to ground. Conversely, the effect on the feedback factor and amplifier load capacitance will be different for each case.

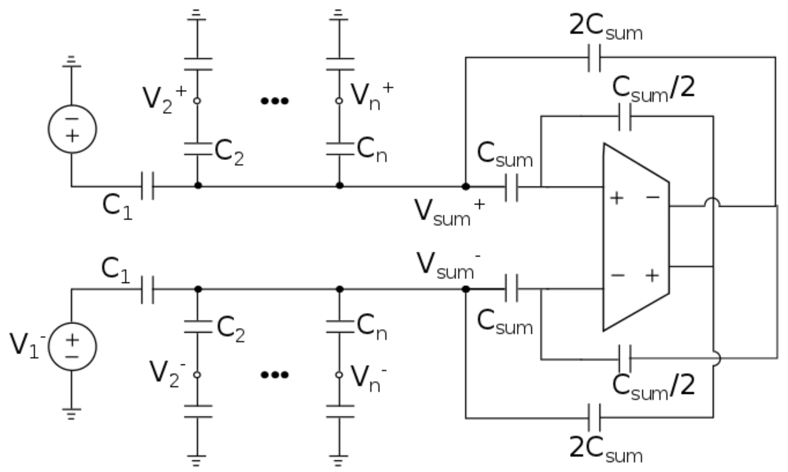

Figure 3.2: Fully differential equality constraint with capacitor values

3.2 Fully Differential Inequality Constraint

Until now we have described the inequality constraint as an equality constraint with an ideal diode. However, in reality this diode is a transmission gate with gate inputs driven by a fully differential comparator. Since we can't compare two differential signals and we don't want to implement a floating voltage source, we can adjust our constraints such that the inequality is enforced on a single differential variable in comparison with zero. This comparison is shown in Fig. 3.3.

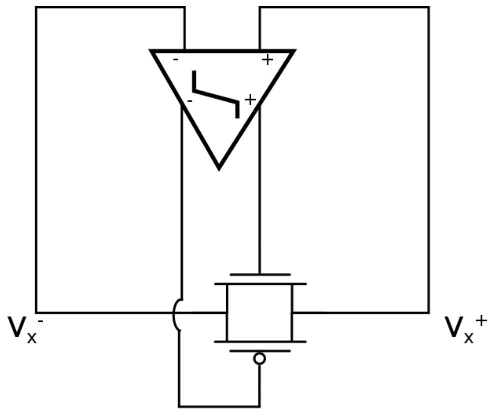

Figure 3.3: Inequality enforcing $V_x \leq 0$. The comparator includes an SR latch to hold the outputs constant during the comparator reset phase.

The figure above enforces the inequality $V_x \leq 0$. However, say we want to solve a problem with nonzero inequality constraints such as (3.3). In this case, we adjust the constraint with a new variable while ensuring that we still solve the same problem. Note that there are additional constraints that would be needed to implement the shown cost function, but we can ignore them for now.

$$\begin{aligned} &min\ x^2 + y^2 \\ &\quad x + 2y + z = 0 \\ &\quad z = -100 \\ &\quad y \leq b \end{aligned} \tag{3.3}$$

The new variable we create is $y' = y - b$ where b is a constant implemented with a voltage source. Then substituting for y we need the circuit to implement the optimization problem.

$$\begin{aligned} &min\ x^2 + y'^2 + 2by' + b^2 \\ &\quad x + 2y' + z + 2b = 0 \\ &\quad z = -100 \\ &\quad y' \leq 0 \end{aligned}$$

We can adjust the redundant constraints as described in Section 2.5 to implement the new cost function. In this case, it's simple as the only additional term is $2by'$. A capacitor between two variables adds their cross term to the cost function; therefore, our new function now has the form shown in (3.4) but will still have the same solution as (3.3). If we wanted to directly solve for y rather than relying on y' for the solution, we can simply add the constraint $y' + 0.5w - y = 0$. Note that this will require an additional amplifier as well as additional redundant constraints to readjust the cost function.

$$\begin{aligned} min\ & x^2 + y'^2 + wy' \\ & x + 2y' + z + w = 0 \\ & z = -100 \\ & y' \leq 0 \\ & w = 2b \\ & y + w \leq \infty \end{aligned} \tag{3.4}$$

Chapter 4

Design Methodology

4.1 System Level Design

4.1.1 Parasitic Capacitance

Because the ability to implement the correct optimization problem depends completely on charge transfer, the accuracy of the solution is highly sensitive to parasitic capacitance. As such, all routing to capacitances has ground shielding and the capacitors are sized to be large enough such that any unaccounted parasitic capacitance won't affect accuracy. Then the only parasitics we are concerned with are the capacitances to the substrate from routing and coupling capacitance to the shielding. These parasitics add extra load capacitance from the variable nodes to ground as well as a large capacitance from the summing node to ground as shown in Fig. 4.1. The summing node capacitance C_{err} is the most problematic parasitic because then the cancelation this circuit is meant to implement is no longer perfect. However, it is simple to account for this with an added capacitance C_{comp} in parallel with the feedback capacitor. The value of C_{comp} is equal to C_{err} in order to correctly compensate for it. Practically, this compensation capacitor would be implemented with a bank of switched-capacitors to allow for tunability. Then as long as the capacitance from the summing node to ground is the only significant parasitic capacitance, the compensation will correct for the error and the circuit will be accurate for multiple input values.

The added load capacitances on variable nodes will change the cost function that the circuit implements. Therefore, they need to be accounted for when determining redundant constraints to implement the cost function. On the other hand, C_{err} will not affect the cost function. It effectively adds a new variable to the constraint which needs to be corrected with C_{comp}. However, because this variable is ground, any cross terms in the cost function created by C_{err} would be multiplied by zero.

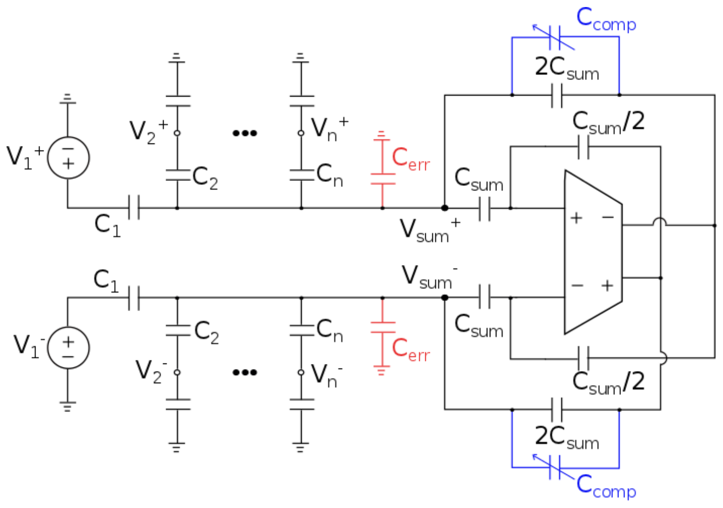

Figure 4.1: Diagram showing addition of compensation capacitor C_{comp} to correct for error due to summing node parasitic capacitance C_{err}.

Both C_{err} and the added variable load capacitances significantly affect the OTA performance as they will change the feedback factor and effective amplifier load capacitance. As such the OTA design may require additional design iterations after layout when parasitics are better known.

4.1.2 Size of Capacitors and Switches

The feedback capacitors need to be large enough such that any unaccounted parasitics will not change the function of the constraint block. They should also be small enough such that the RC time constant from the switches will not be larger than the speed of the amplifier. Likewise, the switches should be small enough such that their drain and source capacitances don't affect the result of the circuit and large enough that the on resistance results in a small enough RC time constant.

4.2 OTA Design

4.2.1 Topology

The OTA used in all the constraint blocks is a fully differential two stage telescopic cascode amplifier as shown in Fig. 4.2. Since both accuracy and speed of the system are important, we needed an architecture with high gain. To be able to handle a wider variety of constraints and therefore dynamic range ratios, we also want high output swing. This allows for more freedom when choosing capacitance values and redundant constraints.

Since we're already using a switched-capacitor configuration with our feedback capacitors, it's simplest to use a switched-capacitor common mode feedback (CMFB) as well. In

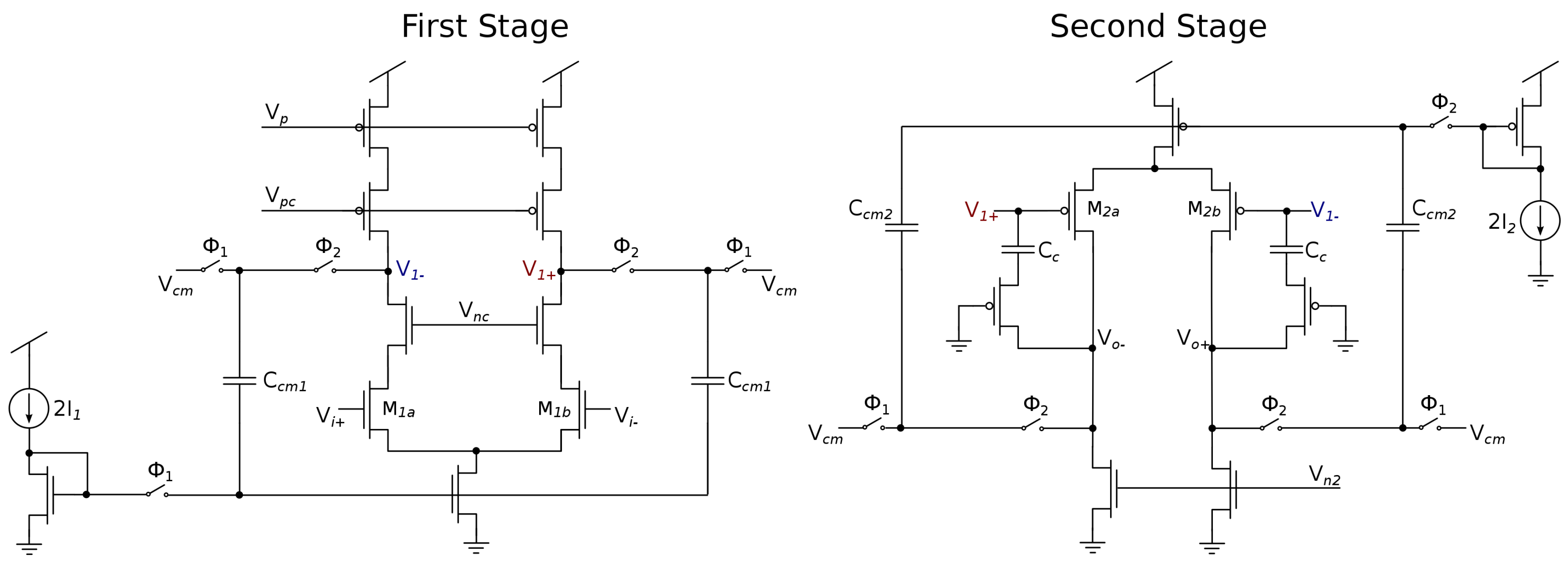

Figure 4.2: Fully differential two stage telescopic cascode amplifier with common mode feedback. Cascode biasing is implemented with wide swing current mirrors.

this configuration, two capacitors sense the common mode of the differential output and feed it back to the tail of the amplifier. Thus for large common mode signals, the tail current increases causing the output common mode to decrease. During the reset mode, we set the initial voltages based off the desired common mode and the desired tail current. For our amplifier we have two CMFB loops, one for each stage. As a switched-capacitor implementation the common mode feedback doesn't add significant poles or zeros and simply shares the existing differential poles and zeros. Since there is a separate common mode loop for each stage and each stage has one dominant pole, the circuit should consequently maintain common mode stability and our only concern will be the loop gain.

4.2.2 Methodology

Given an optimization problem with speed and accuracy specifications, the amplifier for each implemented constraint can be designed using the same methodology. The first thing to determine is the unit size (C_{unit}) for the constraint capacitors. For small capacitance there will be larger noise contributions and the accuracy of the system is more sensitive to parasitics. However, larger capacitors will load the amplifier and require more power. In our case parasitics dominate for the minimum capacitor requirement.

Given the constraint implemented by the amplifier and the effective load capacitances on the optimization variables, we can lump the capacitors seen by the amplifier as in Fig. 4.3. The capacitor C_{sum} is the sum of the constraint capacitors and C_{net} is the effective capacitance seen looking out from the summing node. Since a constraint can be implemented with more than one variable being driven by voltage sources, we generalize the equations assuming there are n optimization variables loaded by an effective capacitance $C_{i,load}$ and m optimization variables driven by voltage sources. Then C_{sum} and C_{net} can be written in the form shown in (4.1), where the C_i's connect the variable nodes to V_{sum} and the C_s's connect the voltage sources to V_{sum}.

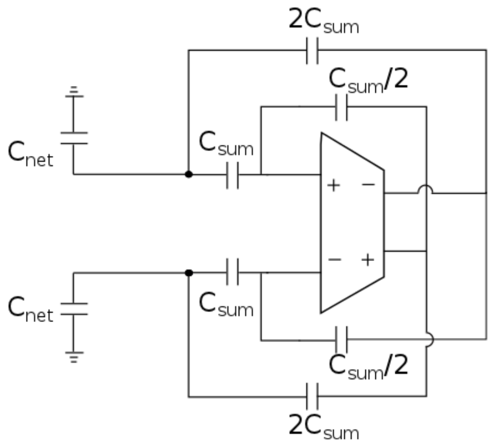

Figure 4.3: Simplified model of constraint capacitances for the purpose of deriving the feedback factor and effective load capacitance

$$C_{sum} = \sum_{i=1}^{n} C_i + \sum_{s=1}^{m} C_s \tag{4.1}$$
$$C_{net} = \sum_{i=1}^{n} \frac{C_i}{1+\frac{C_i}{C_{i,load}}} + \sum_{s=1}^{m} C_s$$

With these lumped capacitances, we can solve for the feedback factor and effective load capacitance at the output of the amplifier. As seen from (4.2), the stability of the system is dependent on $C_{sum} > C_{net}$. This is a result of the amplifier having both positive and negative feeback paths and should be a consideration when choosing redundant constraints.

$$F = \frac{C_{sum}(C_{sum} - C_{net})}{C_{sum}(7C_{sum} + 3C_{net}) + C_{in}(6C_{sum} + 2C_{net})} \tag{4.2}$$

$$C_{L,eff} = \frac{C_{sum}(4C_{sum}^2 + 7C_{sum}C_{in} + 7C_{sum}C_{net} + 5C_{in}C_{net})}{7C_{sum}^2 + 6C_{sum}C_{in} + 3C_{sum}C_{net} + 2C_{in}C_{net}}$$

In (4.3) the equations shown above are expanded to include the effect of the summing node parasitic capacitance (C_{err}) and the compensation capacitance (C_{comp}). Then the stability requirement becomes $C_{sum} + C_{comp} > C_{net} + C_{err}$. Considering that C_{comp} should be roughly equal to C_{err} to maintain solution accuracy, the feedback factor simplifies to (4.4). Since the numerator is the same as in (4.2), C_{comp} will serve to maintain stability as well as accuracy. However, the amplitude of the feedback factor is still reduced, requiring design iterations after the values of C_{err} and C_{comp} are calculated from the layout parasitics.

$$F = \frac{C_{sum}(C_{sum} + C_{comp} - C_{net} - C_{err})}{C_{sum}(7C_{sum} + 3C_{net} + 3C_{comp} + 3C_{err}) + C_{in}(6C_{sum} + 2C_{net} + 2C_{comp} + 2C_{err})} \tag{4.3}$$

$$F = \frac{C_{sum}(C_{sum} - C_{net})}{C_{sum}(7C_{sum} + 3C_{net} + 6C_{comp}) + C_{in}(6C_{sum} + 2C_{net} + 4C_{comp})} \tag{4.4}$$

After calculating F and $C_{L,eff}$ based off an initial guess for C_{in}, we calculate the capacitive loading on the first and second stages, C_{L1} and C_{L2} respectively. For the first iteration we assume there is no self-loading from the transistors and make initial guesses for the common mode capacitors, C_{cm1} and C_{cm2}. In the remaining iterations we obtain these values from the transistor width. The pc, nc, $p2$, and $n2$ labels in (4.5) represent the cascode pmos, cascode nmos, second stage pmos, and second stage nmos respectively.

$$C_{L1} = C_{cm1} + cdd_{pc} + cdd_{nc} + cgg_{p2} \tag{4.5}$$
$$C_{L2} = C_{L,eff} + C_{cm2} + cdd_{p2} + cdd_{n2}$$

Output swing and gain requirements are used to choose initial $V^* = \frac{2I_D}{g_m}$ and transistor length L values respectively. The required output swing depends on the ratio between variable capacitances and effective load capacitances whereas the gain requirement is derived from the static error. As previously mentioned, C_{unit} is already chosen to be large enough that the solution is unaffected by coupling capacitance. Therefore, as long as the miller capacitance (C_c) is large enough to not be a dominant noise source, we can choose its value to minimize power. For example, a low C_c reduces required power in the first stage but increases power in the second stage.

Using the initial values for C_c and F, we calculate the transconductance of the first stage (g_{m1}) from the settling time requirement as in (4.6). The equation for τ originates from the crossover frequency of the closed loop response and is simplified using an approximation for the first pole. Since there are now values for g_{m1} and the V^* of the input device, we can calculate the current and transistor widths for the first stage.

$$\begin{aligned} t_s &= -ln(\epsilon_{dyn})\tau \\ \tau &= \frac{C_c}{Fg_{m1}} \end{aligned} \tag{4.6}$$

We want to place our second pole such that the closed loop response has a phase margin large enough to limit the ripples in the transient step response. Thus we calculate the g_{m2} that satisfies (4.7). In general, $K = 2$ results in a close to optimal transient step response in terms of settling time. Using the value for g_{m2}, we can calculate the second stage current and widths.

$$\begin{aligned} w_{p2} &= \frac{g_{m2}}{\frac{C_{L1}C_{L2}}{C_c} + C_{L1} + C_{L2}} \\ w_{p2} &= Kw_c = KF\frac{g_{m1}}{C_c} \end{aligned} \tag{4.7}$$

We calculate values for the common mode capacitors from the common mode loop gain in (4.8). For a large C_{cm}, we will have a larger gain and a more accurate common mode output but we will also have large loading which will require more power to meet bandwidth requirements. For the first stage we require higher accuracy since the output biases the second stage and therefore we choose a larger loop gain to calculate C_{cm1}.

$$T_{cm} = \frac{4C_{cm}}{2C_{cm} + C_{gs,tail}} A_{dm} \frac{V_{in}^*}{V_{tail}^*} \tag{4.8}$$

With sizing determined we have values for the intrinsic capacitors and the common mode feedback capacitors. Therefore, we can recalculate F, C_{Leff}, C_{L1}, and C_{L2} and reiterate until these values remain constant between iterations. Furthermore, choosing L allows for a larger iteration if the amplifier gain is not large enough. To summarize, the bullet points below show the methodology flow.

1. Choose C_{unit}
2. Calculate initial F, C_{Leff}
3. Guess C_{L1} and C_{L2}
4. Choose V^* and L values
5. Choose C_c
6. Calculate g_{m1} from settling time
7. Calculate g_{m2} from phase margin goal
8. Find sizing
9. Calculate C_{cm1} and C_{cm2}
10. Recalculate F, C_{Leff}, C_{L1}, and C_{L2} - reiterate from 6.
11. Recalculate with new L value if desired - reiterate from 6.

Once the design is finalized, we bias the circuit with current mirrors. In the case of the cascode devices, we use a wide swing current mirror. We also move the zero created by the miller capacitor from the right half plane into the left half plane by adding a transistor in series with the miller capacitor which satisfies $g_{ds} < g_{m2}$ as seen from the equation (4.9).

$$w_z = \frac{1}{C_c(\frac{1}{g_{m2}} - \frac{1}{g_{ds}})} \tag{4.9}$$

4.3 Comparator Topology

For the comparator we chose a strong-arm latch shown in Fig. 4.4 because it offers a better output swing than other designs, such as the CML latch. Since the inequality constraint must be continuously updated, the comparator needs to be clocked at a much faster rate than the rest of the circuit. One consequence of using a regenerative latch is that the required reset state will slightly increase the settling time of the solution.

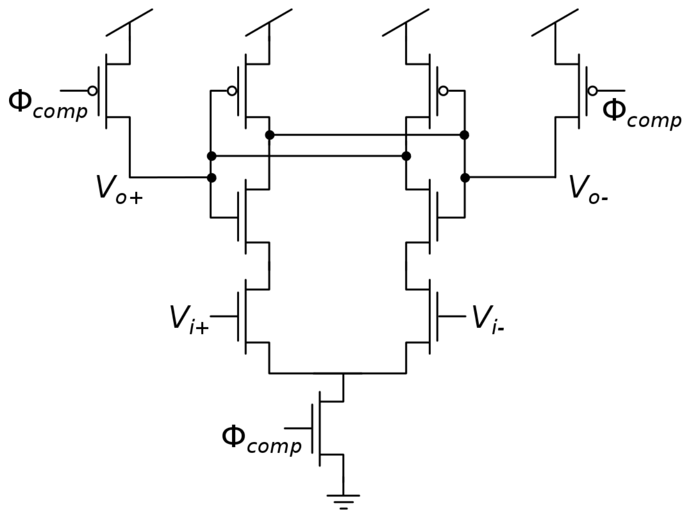

Figure 4.4: Strong-arm latch

4.4 Simulation and Results

The example we implemented is a simple 3 variable equality constraint $V_x + 2V_y + 0.5V_z = 100mV$, the layout for which is shown in Fig. 4.5. We used mom capacitors with a unit capacitance of 75 fF, so the smallest capacitance is 150 fF. Because the parasitic capacitance from each variable to ground is significant, we did not implement load capacitances in the array and simply relied on the coupling capacitance to ground shielding and capacitance from routing to the substrate. The resulting capacitances after extraction are shown in Table 4.1 below.

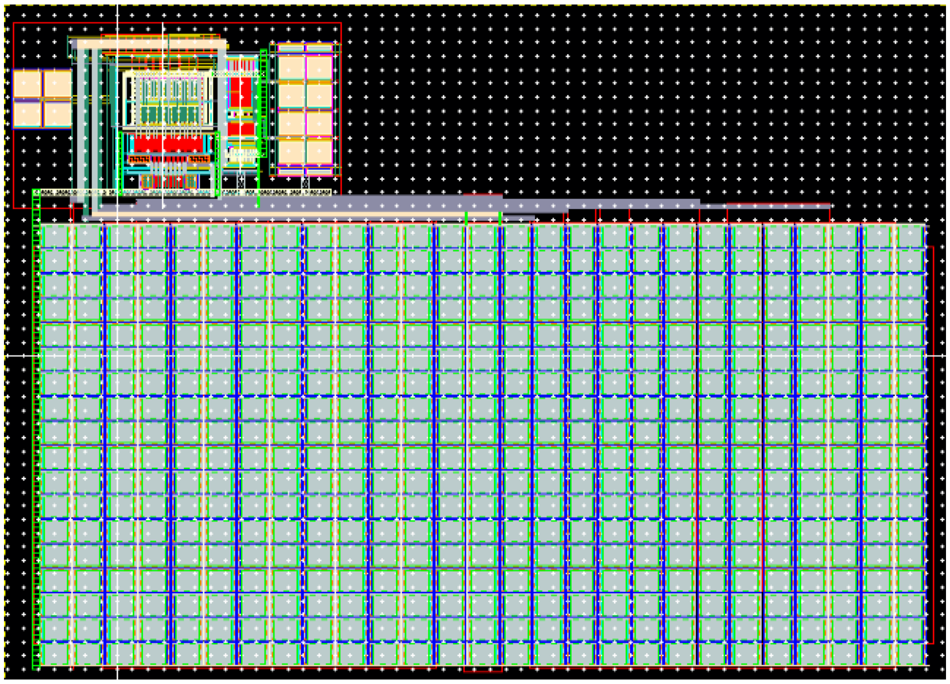

Figure 4.5: Layout of 3 variable QP implemented in TSMC 65nm. Total size is 240μm x 300μm.

Table 4.1: Absolute and Normalized Capacitor Values

Capacitor	Value	Ratio to C_s
C_x	298.7 fF	1
C_y	597.5 fF	2
C_z	149.6 fF	0.5
C_s	298.7 fF	1
$C_{x,load}$	132.7 fF	0.444
$C_{y,load}$	167.2 fF	0.56
$C_{z,load}$	87.3 fF	0.292

The table also includes the ratio of the capacitors to C_s in order to normalize their values and find the implemented optimization problem. The constraints enforced by the circuit are shown in (4.10), where (4.10a) is the main constraint we're implementing, (4.10b)-(4.10d) are redundant inequality constraints set by the load capacitances, and (4.10e) and (4.10f) are constant values set by voltage sources and ground respectively. It should be noted that the variables in these equations are differential.

$$C_xV_x + C_yV_y + C_zV_z + C_sV_s = 0 \tag{4.10a}$$
$$2C_{x,load}V_x + 2C_{x,load}r \leq \infty \tag{4.10b}$$
$$2C_{y,load}V_y + 2C_{y,load}r \leq \infty \tag{4.10c}$$
$$2C_{z,load}V_z + 2C_{z,load}r \leq \infty \tag{4.10d}$$
$$V_s = -100mV \tag{4.10e}$$
$$r = 0 \tag{4.10f}$$

The constraint matrix is determined from the normalized capacitance values in the constraints as shown below. Then the implemented cost function is determined from calculating Q_A as decribed in Section 2.4.

$$A = \begin{bmatrix} A_{eq} \\ A_{ineq} \end{bmatrix} = \begin{bmatrix} \frac{C_x}{C_s} & \frac{C_y}{C_s} & \frac{C_z}{C_s} & 1 & 0 \\ 2\frac{C_{x,load}}{C_s} & 0 & 0 & 0 & 2\frac{C_{x,load}}{C_s} \\ 0 & 2\frac{C_{x,load}}{C_s} & 0 & 0 & 2\frac{C_{x,load}}{C_s} \\ 0 & 0 & 2\frac{C_{x,load}}{C_s} & 0 & 2\frac{C_{x,load}}{C_s} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

After calculating Q_A and substituting $-100mV$ into the cost function for V_s, the final cost function is

$$\frac{1}{1125}(2749x^2 + 3760y^2 + 1657z^2 - 2000xy - 500xz - 1000yz + 100000x + 200000y + 50000z)$$

As a result, we expect the differential values $V_x = 26.93$ mV, $V_y = 30.38$ mV, and $V_z = 24.61$ mV. The extracted results without design iterations after layout are shown in Fig. 4.6. Looking at these results, the variables solve the equality constraint within $\pm 1\%$ error in 50 ns. This is done with a power consumption of 4.32 mW.

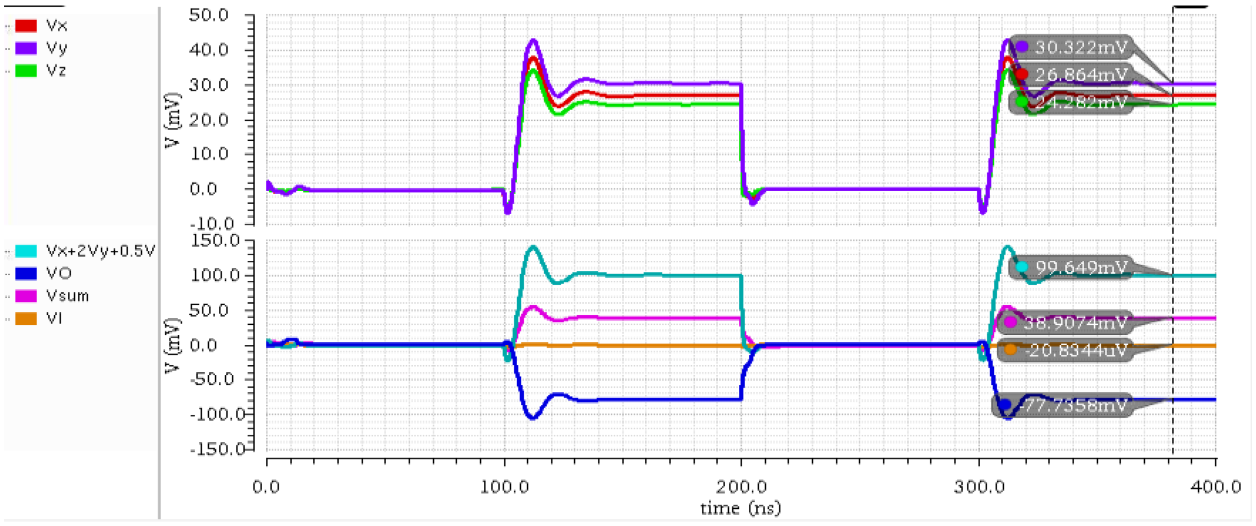

Figure 4.6: Extracted results showing settling time of 50 ns for the equality constraint $V_x + 2V_y + 0.5V_z = 100mV$ with a switching period of 200 ns. The differential values of the summing node and amplifier input and output are also shown.

Sweeping the input changes the equality constraint and is a good check for unaccounted errors due to coupling capacitance for example. Fig. 4.7 shows the resulting $V_x+2V_y+0.5V_z$ waveforms which all settle within $\pm1\%$ of the input voltage. This indicates that there are no significant errors apart from the parasitic capacitance at the summing node which is accounted for by the compensation capacitor.

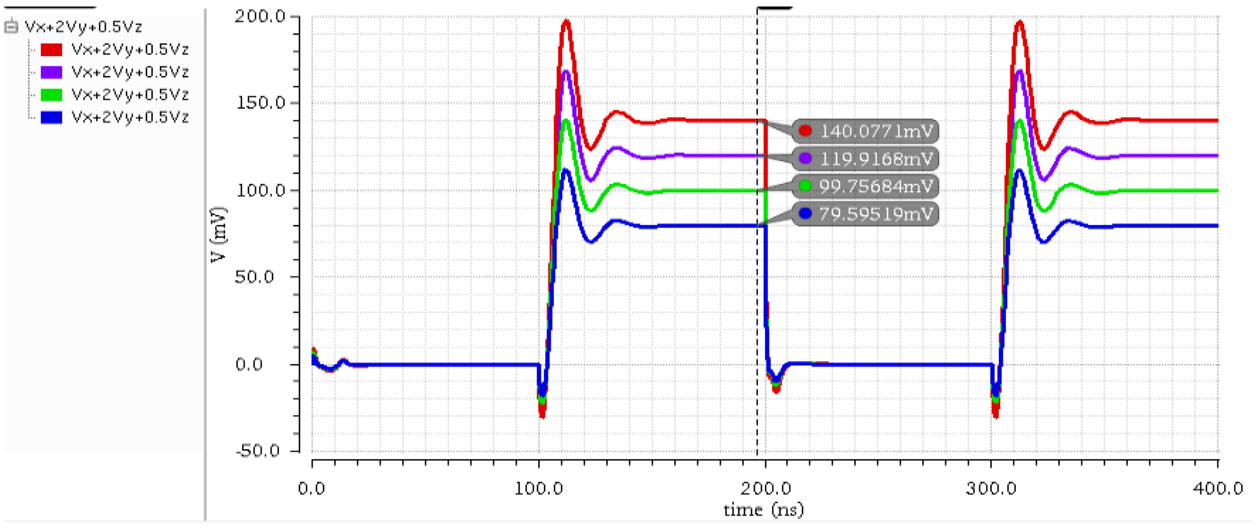

Figure 4.7: Extracted results showing accuracy for the equality constraint $V_x+2V_y+0.5V_z = V_s$ for V_s equal to 80 mV (blue), 100 mV (green), 120 mV (purple), and 140 mV (red).

Chapter 5

Conclusion

This report presented a new method for solving convex optimization problems with an analog circuit. We have proven through charge analysis that the proposed circuit imposes equality constraints. Likewise, we demonstrated that the circuit satisfies KKT conditions and therefore implements a cost function. Furthermore, we described how to modify the circuit in order to implement the desired cost function as well as inequality constraints. Finally, we reviewed the design methodology used for amplifiers and capacitor sizing in the circuit. The results from extracted simulations show that we can solve a simple 3 variable QP with 99% accuracy in 50 ns while consuming 4.32 mW of power. To our knowledge, this is faster than any other reported results.

Considering that the solution speed should improve with further design iterations after layout, our results are promising. Additionally, power consumption can potentially improve, as power conservation methods weren't explored in this work. Further research on this proposed circuit can also include implementing more complex problems to determine latency scaling and fabricating and testing a chip.

Bibliography

[1] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[2] L. Chua and Gui-Nian Lin. Nonlinear programming without computation. *Circuits and Systems, IEEE Transactions on*, 31(2):182 – 188, feb 1984.

[3] Jack B. Dennis. *Mathematical programming and electrical networks*. Technology Press of the Massachusetts Institute of Technology [Cambridge], 1959.

[4] CVX Research Inc. Cvx: Matlab software for disciplined convex programming, March 2015. http://cvxr.com/cvx/.

[5] J.L. Jerez, P.J. Goulart, S. Richter, G.A. Constantinides, E.C. Kerrigan, and M. Morari. Embedded predictive control on an fpga using the fast gradient method. In *Control Conference (ECC), 2013 European*, pages 3614–3620, July 2013.

[6] A. Kelman and F. Borrelli. Parallel nonlinear predictive control. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pages 71–78, Oct 2012.

[7] M.P. Kennedy and L.O. Chua. Neural networks for nonlinear programming. *Circuits and Systems, IEEE Transactions on*, 35(5):554 –562, may 1988.

[8] S. Mariéthoz, U. Mäder, and M. Morari. High-speed fpga implementation of observers and explicit model predictive controllers. In *IEEE IECON, Industrial Electronics Conf.*, Porto, Portugal, November 2009.

[9] J. Mattingley and S. Boyd. Real-time convex optimization in signal processing. *Signal Processing Magazine, IEEE*, 27(3):50–61, May 2010.

[10] P. Patrinos and A. Bemporad. An accelerated dual gradient-projection algorithm for linear model predictive control. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 662–667, Dec 2012.

[11] D. Sheingold. *Nonlinear Circuits Handbook*. Analog Devices, Inc., 1974.

[12] D. Tank and J. Hopfield. Simple 'neural' optimization networks: An a/d converter, signal decision circuit, and a linear programming circuit. *Circuits and Systems, IEEE Transactions on*, 33(5):533 – 541, may 1986.

[13] Sergey Vichik and Francesco Borrelli. Solving linear and quadratic programs with an analog circuit. *Computers & Chemical Engineering*, 70:160–171, 2014.

[14] P. Zometa, Markus Kogel, T. Faulwasser, and R. Findeisen. Implementation aspects of model predictive control for embedded systems. In *American Control Conference (ACC), 2012*, pages 1205–1210, June 2012.