УДК 004.94 DOI 10.31114/2078-7707-2018-1-125-132

Оптимизационные процедуры в системе проектирования аналоговых схем

А.А. Лялинский

Институт проблем проектирования в микроэлектронике РАН, г. Москва

ippm@ippm.ru

***Аннотация* — Оптимизация аналоговых схем на уровне дискретных компонентов является достаточно затратной по времени процедурой (из-за необходимости проведения большого количества Spice-моделирований) с зачастую непредсказуемыми результатами поиска (вследствие сильно выраженной нелинейности исследуемого гиперпространства). В данной работе рассматривается подсистема оптимизации, являющаяся частью АРС (системы проектирования аналоговых схем) и имеющая в своем составе набор процедур, различающихся по скорости работы и точности получаемого решения. Пользователь сам решает, что ему важнее – быстро получить приблизительное решение проблемы или затратить время на получение более точного решения. Использование этих процедур позволяет повысить качество проектируемых схем за счет нахождения решения в наилучшей степени соответствующего поставленным условиям работы схемы.**

***Ключевые слова* — облачные технологии, система схемотехнического проектирования, информационно-поисковая система, оптимизация схем, веб-доступ, веб-сайт.**

I. Введение

Работы, связанные с построением систем оптимизации аналоговых схем, появились в конце 80-х годов [3, 4] – с тех пор, когда компьютеры приобрели мощь, позволяющую проводить достаточно длительные расчеты, причем уже тогда, учитывая наличие некоторой неопределенности с получением желаемого результата, старались создавать системы с диалоговым интерфейсом. С тех пор и до наших дней по этой тематике появилось достаточно много работ [5-12]. Тем не менее оптимизация аналоговых схем имеет ограниченное распространение в силу больших временных и ресурсных затрат на нее, обусловленных необходимостью расчета большого количества вариантов схемы на уровне дискретных компонентов. В то же время эффект от нее (как и от любой другой оптимизационной процедуры) очевиден – мы получаем проект более высокого качества.

В данной работе описывается система оптимизации, входящая в состав системы проектирования аналоговых схем АРС. Всего в АРС планируется реализовать линейку из трех оптимизационных процедур, различающихся между собой скоростью поиска и точностью получаемого решения (см. рис. 1). Это:

- поиск оптимального решения в базе данных,
- поиск перебором на сетке данных,
- градиентный (аналитический) поиск.

На данный момент реализованы первый два метода (поиск в базе данных (БД) и перебор на сетке), которые и будут рассмотрены в данной статье. Градиентный метод планируется реализовать позднее.

Рис. 1. Набор поисковых процедур в АРС

Можно отметить две особенности данной работы:

- наличие нескольких процедур оптимизации, различающихся по точности нахождения решения и скорости поиска;
- реализация данной системы на основе облачных технологий в виде веб-сайта.

Далее в части II рассматриваются вопросы, связанные с нахождением наилучшего решения в базе данных (БД), в части III особенности реализации поиска оптимума методом перебора, в части IV особенности реализации системы в целом в виде веб-сайта.

II. Поиск решений в базе данных

В АРС реализована достаточно уникальная для систем проектирования интегральных схем опция – сохранение результатов расчета схемы в БД с возможностью последующего поиска в ней нужного

решения. При таком подходе к построению системы проектирования сравнительно долгий этап наполнения БД знаниями (образцами схем с различными вариантами параметров составляющих ее элементов и соответствующих им показателям схемы), связанный со Spice-моделированием, отделяется от быстрого этапа непосредственного проектирования схемы путем поиска оптимального решения в БД. Кроме того, экономия времени возможна и в том случае, когда наполнение БД проводится не разработчиком схемы, а иным лицом.

Наполнение БД информацией рассмотрено в статьях, посвященных характеризации схем в АРС. В этой статье основное внимание уделено организации поиска решений.

A. *Структура записей в базе данных*

В БД для хранения решений имеется два типа записей – информация о схеме (табл. 1) и информация о вычисленной точке (табл. 2) (в указанных таблицах показаны только поля, имеющие отношение к хранению решений).

Пользователь, находясь на странице поиска, имеет возможность указания информации в пяти различных блоках, наиболее интересные из них - блок учета характеристик схем (рис. 2) и блок задания поисковых параметров (рис. 3).

Таблица 1

Структура записи с информацией о схеме

ID схемы	netlist	характеристики схемы				
		быстродействующая	мало-шумящая	радиационно-стойкая	прецизионная	…

Таблица 2

Структура записи с информацией о вычисленной точке

ID записи	ID схемы	схемные параметры	вычисленные показатели

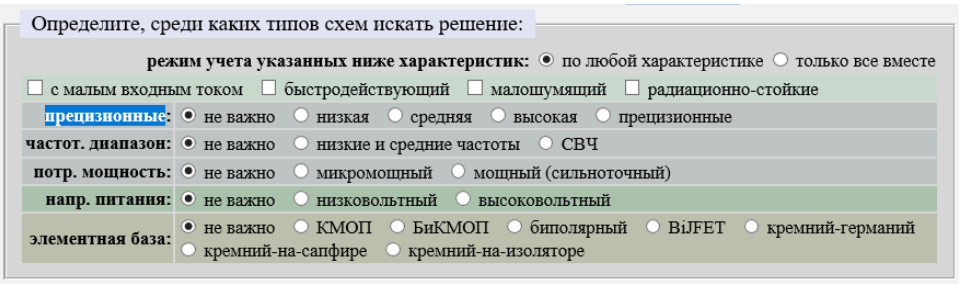


Рис. 2. Блок учета характеристик схем

Рис. 3. Блок задания поисковых параметров

При задании поисковых параметров возможны три варианта указания границ поиска: (1) установка диапазона попадания, задание (2) нижней или (3) верхней границы.

B. *Алгоритм выборки данных*

После получения поискового запроса от пользователя и проверки его корректности происходит выборка записей, имеющих имена параметров, совпадающих с указанными в запросе. Далее определяется, насколько выбранные записи численно

удовлетворяют условиям выборки. Используется следующий алгоритм.

1) Постановка задачи

Найти все записи, удовлетворяющие условию (**n** – число функций):

частный_запрос_1 AND ... AND частный_запрос_n

где **частный_запрос_i** сформулирован в одной из трех форм (I = 1, …, n):

- $f_i^{min} \le f_i \le f_i^{max}$
- $f_i \ge f_i^{min}$
- $f_i \le f_i^{max}$

В результате выполнения запроса мы имеем **m** найденных в базе данных записей, каждая запись – это вектор размерностью **n**. Содержимое i-ого компонента вектора – это структура из двух полей:

- величина i-ой функции, полученная из БД,
- 0 / 1 – признак: выполнено или нет условие по i-ой функции.

В конце работы алгоритма необходимо:

1. **Отсортировать** найденные записи по критерию качества: в начале списка «лучшие», в конце - «худшие».
2. Связать с каждой записью **показатель качества**, позволяющий оценить насколько далеко расположена найденная точка от требуемого решения.

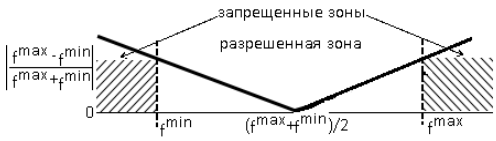
2) Типы ограничений

Формулы для вычисления **показателя качества** q_i подобраны таким образом, что q_i всегда положителен. Наилучшее значение функции соответствует $q_i = 0$. Значение функции всегда нормируется делением полученного значения на свое же константное значение (обычно это величина границы). Это позволяет в дальнейшем суммировать между собой разнородные физические величины.

a) Указан диапазон значений

Если указан диапазон значений $[f_i^{min}, f_i^{max}]$, то **показатель качества** q_i вычисляется следующим образом:

$$q_i = \begin{cases} \left|\dfrac{\mathbf{2} * \boldsymbol{f_i}}{\boldsymbol{f_i^{max}} + \boldsymbol{f_i^{min}}} - \mathbf{1}\right|, \text{если } |f_i^{max} + f_i^{min}| > 10^{-6} \\ \left|\dfrac{\mathbf{2}(\boldsymbol{f_i} + \boldsymbol{f_i^{max}} - \boldsymbol{f_i^{min}})}{\mathbf{3}\boldsymbol{f_i^{max}} - \boldsymbol{f_i^{min}}} - \mathbf{1}\right|, \text{если } |f_i^{max} + f_i^{min}| < 10^{-6} \end{cases}$$

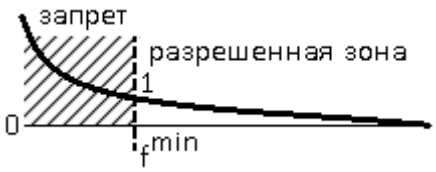


Минимум q_i (равный 0) – в середине диапазона, при движении в любую сторону значение возрастает.

b) Указана нижняя граница

Для условия $f_i \ge f_i^{min}$ полагаем $f_i^{\text{rp}} = f_i^{min}$. **Показатель качества** q_i вычисляется следующим образом:

$$q_i = \begin{cases} e^{1-\frac{f_i}{f_i^{\text{rp}}}}, \text{если } |f_i^{\text{rp}}| > 10^{-6} \\ \dfrac{\pi}{2} - \boldsymbol{arctg(f_i)}, \quad \text{если } |f_i^{\text{rp}}| \le 10^{-6} \end{cases}$$

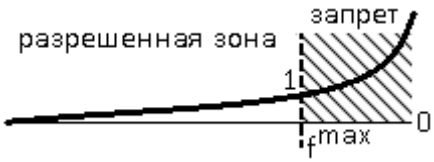


Минимум q_i (стремящийся к 0) – в положительной бесконечности; значение в граничной точке: 1; за пределами заданного полупространства значение растет от 1 до бесконечности.

c) Указана верхняя граница

Для условия $f_i \le f_i^{max}$ полагаем $f_i^{\text{rp}} = f_i^{max}$. **Показатель качества** q_i вычисляется следующим образом:

$$q_i = \begin{cases} e^{\frac{f_i}{f_i^{\text{rp}}}-1}, \text{если } |f_i^{\text{rp}}| > 10^{-6} \\ \dfrac{\pi}{2} + \boldsymbol{arctg(f_i)}, \quad \text{если } |f_i^{\text{rp}}| \le 10^{-6} \end{cases}$$



Минимум q_i (стремящийся к 0) – в отрицательной бесконечности; значение в граничной точке: 1; за пределами заданного полупространства значение растет от 1 до бесконечности.

3) Сортировка записей по качеству

a) Критерии, используемые при сортировке

Сортировка выполняется по трем критериям. Для этого для каждой записи вычисляются следующие значения:

1. n^{sat} - количество удовлетворенных условий.

2. $\sum_{i=1}^{n_{sat}} q_i^{sat}$ – сумма показателей качества удовлетворенных условий.

3. $\sum_{i=1}^{n_{unsat}} q_i^{unsat}$ – сумма показателей качества неудовлетворенных условий (n_{unsat} - количество неудовлетворенных условий, n_{unsat} = n - n^{sat}).

b) Алгоритм сортировки

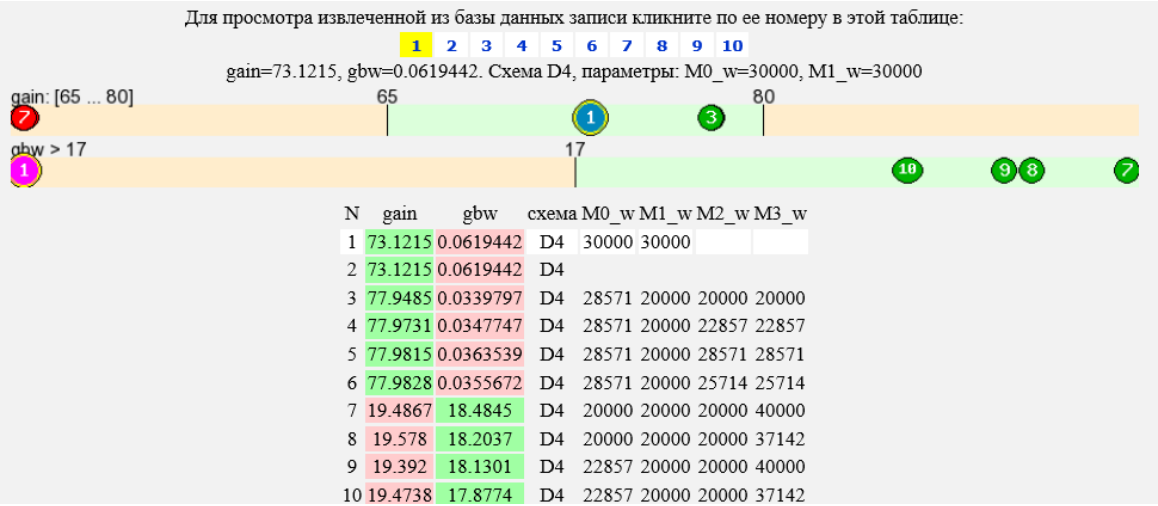
1. Сравнить записи по критерию 1. Если не равны, то запись с большим n^{sat} - лучшая.

2. Иначе («равны по критерию 1») сравнить записи по критерию 2. Если не равны, то запись с меньшей суммой лучшая.

3. Иначе («равны по критериям 1 и 2») сравнить записи по критерию 3. Если не равны, то запись с меньшей суммой лучшая.

4. Иначе («равны по всем критериям») записи неразличимы.

Данный алгоритм позволяет не только определить, удовлетворяет ли поставленным условиям та или иная запись, но и численно охарактеризовать, насколько хорошо точка решения, представленная данной записью, удовлетворяет поставленным условиям. Результатом работы данного алгоритма является отранжированный по условию «лучший-худший» список, в котором учтены все заданные пользователем условия. На рис. 4 показан пример выдачи результатов поиска. Задан поиск точек, имеющих коэффициент усиления в диапазоне [65 … 80] и частоту среза не менее 17 МГц. Показаны первые 10 наилучших точек. Зеленым отмечено попадание в заданный диапазон или интервал. Красным – выход за заданные границы.



N	gain	gbw	схема	M0_w	M1_w	M2_w	M3_w
1	73.1215	0.0619442	D4	30000	30000		
2	73.1215	0.0619442	D4				
3	77.9485	0.0339797	D4	28571	20000	20000	20000
4	77.9731	0.0347747	D4	28571	20000	22857	22857
5	77.9815	0.0363539	D4	28571	20000	28571	28571
6	77.9828	0.0355672	D4	28571	20000	25714	25714
7	19.4867	18.4845	D4	20000	20000	20000	40000
8	19.578	18.2037	D4	20000	20000	20000	37142
9	19.392	18.1301	D4	22857	20000	20000	40000
10	19.4738	17.8774	D4	22857	20000	20000	37142

Рис. 4. Показ результатов поиска по БД

III. Поиск перебором

Данный вид поиска позволяет довольно эффективно находить оптимальные решения при условии, что задано сравнительно небольшое количество варьируемых параметров и известны нижний и верхний пределы их изменений.

Учитывая особенности построения задания на моделирование (так как все точки моделирования известны заранее, то все они считаются за один вызов программы Spice-моделирования) время отклика системы мало несмотря на большое количество проверяемых точек.

Описание алгоритма предваряется постановочной частью.

A. Постановка задачи

Найти оптимальную точку при следующих условиях:

- задан $\overline{X}$ – вектор аргументов размерности n: $\overline{X} = [x_1, x_2, \ldots, x_n]$, по каждому аргументу указаны ограничения на допустимый диапазон изменения;
- имеется набор показателей (частных целевых функций $F = f(f_i), i = 1, 2, \ldots, m$), для каждого показателя указан способ вычисления его оптимального значения.

1) Ограничения на входное пространство

Каждый аргумент имеет допустимый диапазон изменения величины: $x_i = [x_i^{min}, x_i^{max}]$. Нахождение аргумента вне заданного диапазона невозможно. Исключение – заданная пользователем стартовая точка алгоритма.

2) Дискретность входного пространства

По каждой из переменных x_i заранее известно, определена ли она как целочисленная или float-переменная. Для переменных типа float вводится искусственная дискретизация пространства: для данной переменной определяется шаг сетки Δ_i на основании ее диапазона значений и d_i - заранее заданного числа точек в диапазоне: $\Delta_i = \frac{x_i^{max} - x_i^{min}}{d_i}$. Узлы сетки состоят из следующих значений: $x_i = [x_i^{min},\ x_i^{min} + \Delta_i\ x_i^{min} + 2 * \Delta_i, \dots, x_i^{max}]$.

Если в процессе работы алгоритма вычисленное значение аргумента не попадает в узлы сетки, то оно всегда заменяется значением ближайшего узла сетки.

3) Характер требований к частным целевым функциям

Критерий «успеха», т.е. нахождение некоего оптимального (при заданных ограничениях на задачу) решения, для каждой целевой функции должен быть сформулирован одним из следующих способов:

- задан диапазон значений $[\ f_i^{min}, f_i^{max}]$, являющийся оптимальным для данной целевой функции;
- задано граничное значение функции (максимум или минимум, диапазон значений при этом определяется как левое $(\ -\infty, f_i^{max}]$ или правое $[\ f_i^{min}, \infty)$ полупространство);
- указано, что оптимальным будет минимальное или максимальное значение функции на всем множестве просмотренных точек.

4) Нормировка функции

Используется нормировка функций, аналогичная используемой при выборке из БД.

5) Определение наилучшей точки

Полученная на каждом шаге алгоритма точка пространства входных параметров добавляется к накапливаемому множеству точек. Данное множество сортируется по определенному критерию, в результате после каждого шага мы имеем возможность выстроить точки от «лучшей» к «худшей». Сортировка точек производится в 3 этапа:

- вначале отбраковываются точки, имеющие значения функций вне разрешенного диапазона,
- после этого сравниваются их качественные показатели,
- и, если это не дало не результата, то сравниваются количественные характеристики.

Качественное сравнение основано на **ранжировании** значений функций, количественное – на сравнении их норм.

6) Понятие ранга частной целевой функции

Рангом функции (R_i) является целое число в диапазоне от 0 до 2, определяемое по следующим правилам:

0	значение функции **вне** диапазона допустимых значений,
1	значение функции **внутри** диапазона допустимых значений,
2	**наилучшее** значение среди имеющихся на данных момент точек ранга R_1 (определяется по норме функции).

7) Характеристика точки

Точка характеризуется триадой $\{S_0\ ,\ S_1\ ,\ S_2\}$, где S_i – количество i-ых рангов частных целевых функций.

8) Отбраковка точек с недопустимыми значениями функций

Даны две точки a и b, принадлежащие пространству варьируемых параметров. Задача – определить среди них наилучшую. Необходимо проанализировать S_0^a и S_0^b.

1) Если $S_0^a = 0$, а $S_0^b > 0$, то точка «a» наилучшая в этой паре.

2) Иначе если $S_0^a > 0$, а $S_0^b = 0$, то точка «b» наилучшая в этой паре.

3) Иначе если $S_0^a \neq S_0^b$, то точка с меньшим значением S_0 наилучшая в этой паре.

4) Иначе считать точки a и b неразличимыми по критерию разбраковки. Перейти к сравнению по рангу.

9) Сравнение точек по рангу

Даны две точки a и b, принадлежащие пространству варьируемых параметров. Задача – определить среди них наилучшую. Используется следующий алгоритм.

1) Сравнить S_2^a и S_2^b. Если они отличаются, то точка с наивысшим значением S_2 считается наилучшей.

2) Иначе сравнить S_1^a и S_1^b. Если они отличаются, то точка с наивысшим значением S_1 считается наилучшей.

3) Иначе считать точки a и b неразличимыми по рангу. Перейти к сравнению по норме.

10) Сравнение точек по норме

Даны две точки **a** и **b**, принадлежащие пространству варьируемых параметров и неразличимые по рангу (т.е. имеющие одинаковое количество по всем рангам). Задача – определить среди них наилучшую. Используется следующий алгоритм.

1) Для всех точек объединенного ранга R_{12} (т.е. точек, отличных от ранга R_0) сравнить попарно значения норм частных целевых функций $\|f_i^a\|$ и $\|f_i^b\|$. Функция с меньшей нормой получает «балл преимущества» p_i. Точка с большим количеством баллов считается лучшей.

2) Если в предыдущем пункте лучшая точка не выявлена, то проделать аналогичное сравнение для точек ранга R_0.

3) Если оба предыдущих пункта не выявили лучшую точку, то считать точки неразличимыми по норме. Принять точку **a** как лучшую.

11) Критерий останова алгоритма

Алгоритм завершает свою работу при выполнении любого из следующих условий:

- перебраны все возможные дискретные точки в пространстве аргумента,
- в течение некоторого (заранее определенного) количества итераций не наступает улучшения.

При завершении работы алгоритмы выдаются следующие показатели:

- достигнут успех или нет для тех частных функций, для которых указаны показатели качества,
- сравнение значения для каждой функции в исходной точке и в точке с наилучшим значением обобщенного показателя F,
- дискретные значения по каждому аргументу x_i, соответствующие исходной и наилучшей из найденных точек,
- критерий остановки алгоритма.

Далее собственно описание алгоритма.

B. Описание алгоритма оптимизации

Состоит из трех этапов – инициирующий этап с фиксированным числом шагов и последующие поисковые этапы с произвольным числом шагом.

Этап 1 – вычисление целевой функции в угловых точках

1. Вычислить и запомнить вектор аргумента и целевую функцию в исходной точке и во всех угловых точках гиперпространства входных переменных:

$$F_0 = f\left(x_1^{init}, x_2^{init}, x_3^{init}, \dots, x_n^{init}\right),$$
$$F_1 = f\left(x_1^{min}, x_2^{min}, x_3^{min}, \dots, x_n^{min}\right),$$
$$F_2 = f\left(x_1^{max}, x_2^{min}, x_3^{min}, \dots, x_n^{min}\right),$$
$$F_3 = f\left(x_1^{min}, x_2^{max}, x_3^{min}, \dots, x_n^{min}\right),$$
$$F_4 = f\left(x_1^{max}, x_2^{max}, x_3^{min}, \dots, x_n^{min}\right),$$
$$\dots$$
$$F_k = f\left(x_1^{max}, x_2^{max}, x_3^{max}, \dots, x_n^{max}\right)$$

2. Найти среди $F_0, \dots, F_k$ наилучшую точку F_{best}.

3. Положить величину коэффициента длины ребра edgeLength = 0.5.

Этап 2 - вычисление целевой функции в окрестности наилучшей точки

4. Вычислить значения аргумента на точках, расположенных на прямых, соединяющих наилучшую точку с угловыми точками гиперпространства входных переменных. Расстояние от наилучшей точки до вычисляемой – edgeLength% длины данного отрезка:

$$x_i^{m_j} = x_i^{F_j} + edgeLength * \left(x_i^{F_{best}} - x_i^{F_j}\right) -$$
вычисляемая точка на прямой.

5. Если не получено ни одной новой точки (т.е. все получаемые новые точки ранее уже вычислялись), то выход (конец алгоритма).

6. Вычислить целевую функцию в предложенных точках:

$$F_1 = f\left(x_1^{m_1}, x_2^{m_1}, x_3^{m_1}, \dots, x_n^{m_1}\right),$$
$$F_2 = f\left(x_1^{m_2}, x_2^{m_2}, x_3^{m_2}, \dots, x_n^{m_2}\right),$$
$$\dots$$
$$F_k = f\left(x_1^{m_k}, x_2^{m_k}, x_3^{m_k}, \dots, x_n^{m_k}\right),$$

7. Найти среди $F_1, \dots, F_k$ наилучшую точку F_{best}^{try}.

Этап 3 – анализ полученной окрестности

8. Если F_{best}^{try} лучше F_{best}, то принять F_{best}^{try} как новую наилучшую точку.

9. Уменьшить вдвое величину коэффициента длины ребра: edgeLength *= 0.5.

10. Если edgeLength < 1.e-6, то выход.

11. Повторять до остановки алгоритма этапы 2-3.

Этап 4 – выявление наилучших точек

12. Путем просмотра и частичной сортировки всего массива имеющихся точек найти 5 наилучших.

C. Пример расчета

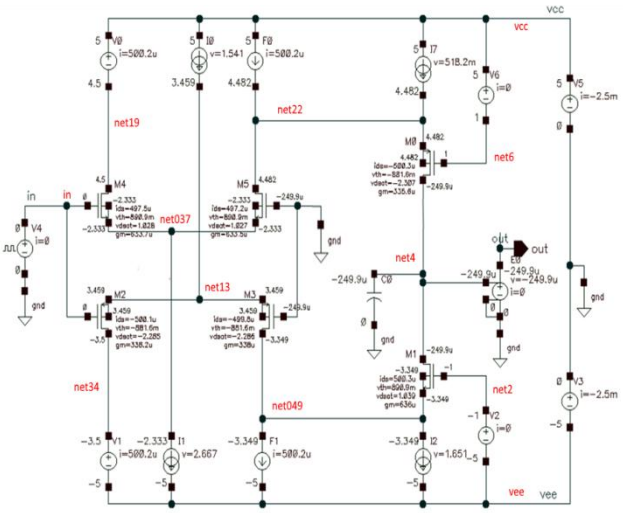


Рис. 5. Схема усилителя, используемого в качестве теста

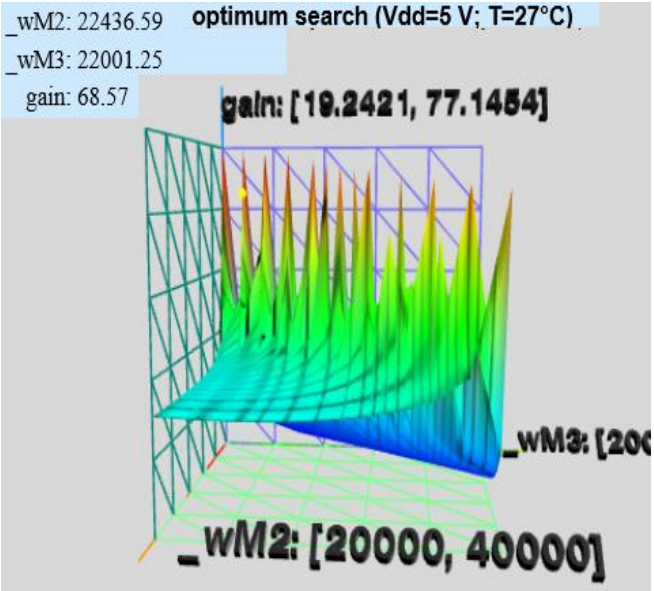


Рис. 6. Поверхность поиска для коэффициента усиления

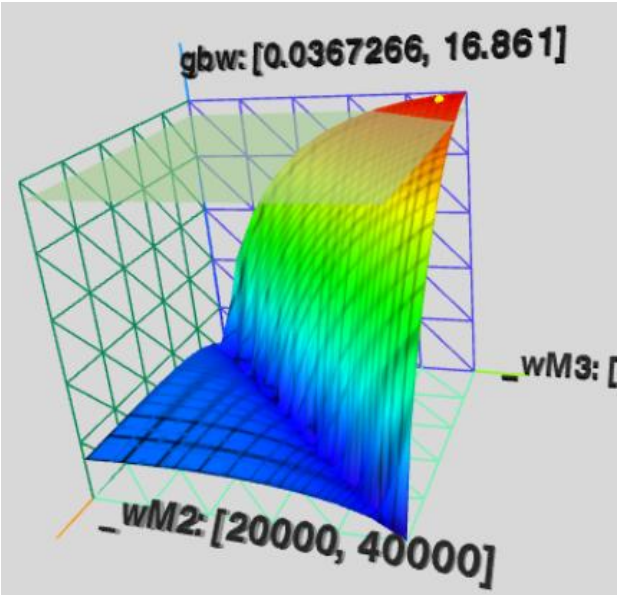


Рис. 7. Поверхность поиска для частоты среза

Для схемы усилителя (6 MOS-транзисторов) в качестве варьируемых параметров заданы ширины транзисторов M2 и M3 в диапазоне [20u-40u] с разбивкой каждого параметра на 10 точек. Ищется максимальный коэффициент усиления и частота среза не ниже 16 МГц. Проверено около 900 точек за 30 сек, результаты представлены на рис. 6 и 7. На рис. 7 полупрозрачная плоскость отмечает заданное пользователем граничное значение.

IV. Особенности реализации

Система АРС, в которой реализован описанный в данный статье функционал, создана на основе облачных технологий. Общение пользователя с системой ведется через обычный веб-сайт. Результаты выдаются в виде веб-страниц в форме графиков и таблиц. Вычислительная часть системы отделена от веб-сервера и реализована на отдельном вычислительном сервере.

V. Выводы

Разработанная и включенная в состав АРС подсистема оптимизации аналоговых схем предоставляет разработчику схем на выбор несколько различающихся по скорости работы и точности нахождения решения процедур. В отличие от существующих систем оптимизации данный проект реализован на основе облачных технологий и представляет собой веб-сайт с подключенным к нему вычислительным сервером, что значительно упрощает работу с системой и расширяет круг доступа к ней.

Литература

[1] Библиотека стандартных ячеек компании Synopsis: https://www.synopsys.com/dw/ipdir.php?ds=dwc_standard_cell (проверено: 08.02.2018).

[2] А.Л. Стемпковский, С.В. Гаврилов, А.Л. Глебов. Методы логического и логико-временного анализа цифровых КМОП СБИС // Российская академия наук, Институт проблем проектирования в микроэлектронике, под общ. ред. А. Л. Стемпковского. Москва, 2007. 224 с.

[3] R. Harjani, R. Rutenbar, L. Richard Carley, OASYS: A Framework for Analog Circuit Synthesis // IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 1989. Vol. 8. No. 12. P. 1247-1266.

[4] Degrauwe M., et al.: IDAC: An interactive design tool for analog CMOS circuits // IEEE J. Solid-State Circuits - 1987. Vol. 22. P. 1106-1115.

[5] Гурарий М.М., Жаров М.М., Русаков С.Г., Ульянов С.Л., Лялинский А.А. Подсистема параметрической оптимизации КМОП операционных усилителей // Проблемы разработки перспективных микро- и наноэлектронных систем (МЭС). 2016. №1. С. 109-116.

[6] Harvey J.P., Elmasry M.I., Leung B.: STAIC: An interactive framework for synthesizing CMOS and BICMOS analog circuits // IEEE Trans. Computer-Aided Design. 1992. 11(11). P. 1402–1417.

[7] Ochotta E.S., Rutenbar R.A., Carley L.R.: Synthesis of high-performance analog circuits in ASTRX/OBLX // IEEE Trans. Computer- Aided Design. 1996. 15(3). P. 273–294.

[8] Gaspar-Cunha A., Vieira A. A. Multi-Objective Evolutionary Algorithm Using Neural Networks To Approximate Fitness Evaluations // Int. J. of Computers, Systems and Signals. 2005. 6(1). P. 18–36.

[9] M.F.M. Barros, J.M.C. Guilherme, N.C.G. Horta. Analog Circuits and Systems Optimization Based on Evolutionary Computation Techniques. Berlin: Springer-Verlag, 2010.

[10] Nye W., Riley D.C., Sangiovanni-Vincentelli A., Tits A.L. DELIGHT.SPICE: An optimization-based system for the design of integrated circuits // IEEE Trans. Computer-Aided Design. 1988. Vol. 7. No 4. P. 501–519.

[11] Sonmez O., Dundar G. Simulation-based analog and RF circuit synthesis using modified evolutionary strategies algorithms // Integration, the VLSI Journal. 2011. 44. P. 144–154.

[12] Nicosia G., Rinaudo S., Sciacca E. An evolutionary algorithm-based approach to robust analog circuit design using constrained multi-objective optimization. // Knowledge-Based Systems. 2008. 21: P. 175-183.

Optimization Procedures in the Analog Circuit CAD System

A.A. Lyalinsky

Institute for Design Problems in Microelectronics of Russian Academy of Sciences, Moscow

ippm@ippm.ru

***Abstract* — Optimization of analog circuits on the level of discrete components is quite time-consuming procedure (due to the need of a large number of Spice-simulations) with often unpredictable search results (due to the strong nonlinearity of the studied hyperspace). In this paper, we consider the optimization subsystem, which is part of the APC (analog circuit design system) and has a set of procedures that are differed by speed and accuracy of the resulting solution. The user decides for himself what is more important to him – to get an approximate solution of the problem quickly or take the time to get a more accurate solution. The usage of these procedures allows improvement of the designed circuits quality by finding the solution with the best corresponding to the working conditions of the circuit.**

***Keywords* — cloud computing, circuit simulation system, information system, analog circuit optimization, web access, website.**

REFERENCES

[1] Synopsis standard cells library: https://www.synopsys.com/dw/ipdir.php?ds=dwc_standard_cell (last accessed: 08.02.2018).

[2] A.L. Stempkovsky, S.V. Gavrilov, A.L. Glebov. Metody logicheskogo i logiko-vremennogo analiza cifrovyh KMOP SBIS // Rossijskaya akademiya nauk, Institut problem proektirovaniya v mikroehlektronike, pod obshch. red. A. L. Stempkovskogo. Moskva, 2007. 224 c.

[3] R. Harjani, R. Rutenbar, L. Richard Carley, OASYS: A Framework for Analog Circuit Synthesis // IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, 1989. Vol. 8. No. 12. P. 1247-1266.

[4] Degrauwe M., et al.: IDAC: An interactive design tool for analog CMOS circuits // IEEE J. Solid-State Circuits - 1987. Vol. 22. P. 1106-1115.

[5] Gourary M.M., Zharov M.M., Rusakov S.G., Ulyanov S.L., Lyalinsky A.A. Parameter optimization subsystem of CMOS operational amplifiers // Problems of Perspective Micro- and Nanoelectronic Systems Development. 2016. №1. P. 109-116.

[6] Harvey J.P., Elmasry M.I., Leung B.: STAIC: An interactive framework for synthesizing CMOS and BICMOS analog circuits // IEEE Trans. Computer-Aided Design – 1992. - 11(11). P. 1402–1417.

[7] Ochotta E.S., Rutenbar R.A., Carley L.R.: Synthesis of high-performance analog circuits in ASTRX/OBLX // IEEE Trans. Computer- Aided Design. 1996. - 15(3). P. 273–294.

[8] Gaspar-Cunha A., Vieira A. A. Multi-Objective Evolutionary Algorithm Using Neural Networks To Approximate Fitness Evaluations // Int. J. of Computers, Systems and Signals. 2005. 6(1). P. 18–36.

[9] M.F.M. Barros, J.M.C. Guilherme, N.C.G. Horta. Analog Circuits and Systems Optimization Based on Evolutionary Computation Techniques. Berlin: Springer-Verlag, 2010.

[10] Nye W., Riley D.C., Sangiovanni-Vincentelli A., Tits A.L. DELIGHT.SPICE: An optimization-based system for the design of integrated circuits // IEEE Trans. Computer-Aided Design. 1988. vol. 7. No 4. P. 501–519.

[11] Sonmez O., Dundar G. Simulation-based analog and RF circuit synthesis using modified evolutionary strategies algorithms // Integration, the VLSI Journal. 2011. 44. P. 144–154.

[12] Nicosia G., Rinaudo S., Sciacca E. An evolutionary algorithm-based approach to robust analog circuit design using constrained multi-objective optimization. // Knowledge-Based Systems. 2008. 21: P. 175-183.