ISBN 5-94052-066-0 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ. ПЕРЕСЛАВЛЬ-ЗАЛЕССКИЙ, 2004

УДК 519.6

А. В. Захаров, В. М. Хачумов

Алгоритмы CORDIC. Современное состояние и перспективы

Аннотация. В работе представлена классификация алгоритмов семейства CORDIC по особенностям их реализации. Описаны основные методики ускорения базовых алгоритмов. Предложен новый подход к распараллеливанию CORDIC-алгоритмов, основанный на технике разрядных вычислений, и приведены результаты численного моделирования вычисления тригонометрических функций на его основе. Изложены обобщения алгоритма на трехмерный случай.

Ключевые слова и фразы: алгоритмы CORDIC, параллельные вычисления, цифровая обработка сигналов.

1. Введение

Несмотря на то, что исторически алгоритмы CORDIC (Coordinate Rotation in a Digital Compute) разработаны достаточно давно, интерес к ним не ослабевает до сих пор. Это объясняется высокими потенциальными возможностями данного семейства алгоритмов с точки зрения реализации крупных операций, широко используемых в цифровой обработке сигналов, при решении систем уравнений, в машинной графике, управлении движением, навигации. Перспективным является любое возможное ускорение базовых CORDIC-операций, что подтверждается большим количеством разнообразных подходов для достижения этой цели. В последнее время получены некоторые новые результаты, которые расширили практические горизонты использования CORDIC-алгоритмов. В частности, представляют интерес техника параллельных вычислений на уровне базовых элементов и расширение функциональных возможностей метода.

Работа выполнена при поддержке Российского фонда фундаментальных исследований (проект РФФИ № 03-01-00808).

2. Современные исследования и классификация CORDIC-алгоритмов

Для того чтобы охватить модификации CORDIC-алгоритмов, ниже предложены две классификационные схемы — по особенностям построения и областям применения с указанием соответствующих работ. С точки зрения реализации модификации CORDIC-алгоритмов могут быть классифицированы следующим образом:

- по основанию счисления операндов:
  - алгоритмы с основанием 4 [5, 6, 9, 21];
  - алгоритмы с большим основанием (до 1024) [8];
- по используемым системам представления чисел:
  - избыточная, в виде цифр со знаком (signed digit); [5, 14, 23, 24, 27];
  - избыточная, с сохранением переноса (carry save) [5, 14, 20, 21, 23];
- по методу коррекции результата:
  - умножение на величину, обратную коэффициенту деформации, после окончания итераций [1, 3, 15];
  - добавление специальных масштабирующих итераций [5, 9, 21, 32];
  - параллельное вычисление корректируемого результата [26];
- по технике ускорения вычислений:
  - конвейеризация вычислений [30, 31];
  - использование предварительной декомпозиции угла поворота [14, 18, 32];
  - алгоритмы с предсказанием значений операторов [7, 17, 23, 32];
  - использование оценки знака по малому числу старших битов [6, 7, 21, 23];
  - алгоритм с ветвлением [13, 22].

Можно отметить, что, несмотря на богатство всевозможных техник, предложенных для ускорения алгоритма, почти все они сохраняют его итерационный характер. Ускорение достигается за счет уменьшения общего количества итераций или требуемого количества аппаратуры. Исключением является, пожалуй, лишь техника с предсказанием значений операторов, однако, полученный алгоритм все равно

не является полностью параллельным. Можно выделить основные области применения CORDIC-алгоритмов:

(1) различные формы преобразования Фурье: БПФ, ДПФ, дискретные синус- и косинус-преобразования [11];
(2) алгоритмы линейной алгебры: преобразования Хаусхолдера (Householder transformations) [16], алгоритм разложения матрицы по сингулярным значениям (SVD) [10, 14], обобщение поворота вектора на многомерный случай [12];
(3) алгоритмы цифровой обработки сигналов: алгоритмы цифровой фильтрации [18] и алгоритмы обработки изображений — преобразование Хоу (Hough transform) [9].

Вопросам сходимости и точности CORDIC-алгоритмов посвящены работы [1, 3, 4, 19, 28, 29].

3. Краткий обзор CORDIC-алгоритмов

Изначально алгоритм CORDIC представлял собой итерационную процедуру для выполнения операции «поворот» векторов на плоскости на произвольный угол, используя только операции сдвига и сложения [28]. Как известно, данное преобразование определяется следующими соотношениями

$$\begin{aligned} x &= x_0 \cos\phi - y_0 \sin\phi \\ y &= x_0 \sin\phi + y_0 \cos\phi \end{aligned} \tag{1}$$

Переписывая равенства (1) в виде

$$\begin{aligned} x &= \cos\phi \cdot (x_0 - y_0 tg\phi) \\ y &= \cos\phi \cdot (y_0 + x_0 tg\phi) \end{aligned} \tag{2}$$

и выбирая такой угол поворота, что $tg\phi = \pm 2^{-i}$, легко заметить, что умножение на тангенс в формулах (2) представляет собой всего лишь операцию сдвига на i разрядов в двоичной системе счисления. Следовательно, если мы представим некоторый произвольный угол в виде суммы углов $\alpha_i = \pm arctg(2^{-i})$, i = 0,1,2... n, то операция поворота может быть разложена в серию последовательно выполненных элементарных поворотов. Отметим, что величина множителя $\cos(\alpha_i) = \cos(-\alpha_i)$ не зависит от направления поворота и является константой на каждом шаге. Таким образом, приходим к следующим итерационным формулам для i-го шага (i=0,1... n):

$$\begin{aligned} x_{i+1} &= K_i \cdot (x_i - \sigma_i y_i 2^{-i}) \\ y_{i+1} &= K_i \cdot (y_i + \sigma_i x_i 2^{-i}) \end{aligned} \tag{3}$$

Здесь $K_i = \cos(arctg(2^{-i})) = \frac{1}{\sqrt{1+\frac{1}{tg^2(arctg(2^{-i}))}}} = \frac{1}{\sqrt{1+2^{-2i}}}$ — коэффициент деформации вектора на i-м шаге $\sigma_i \in \{-1, +1\}$ — оператор, определяющий направление поворота (по часовой стрелке или против часовой стрелки). Исключая величину K_i из (3), получим итерационный алгоритм для операции «поворот», использующий только операции сложения и сдвига.

$$\begin{aligned} x_{i+1} &= x_i - \sigma_i y_i 2^{-i} \\ y_{i+1} &= y_i + \sigma_i x_i 2^{-i} \end{aligned} \tag{4}$$

Произведение коэффициентов деформации для всех шагов является сходящимся и дает суммарное растяжение исходного вектора в $K_n = \prod_{i=0}^{n-1} K_i = \prod_{i=0}^{n-1} \sqrt{1+2^{-2i}} \approx 1.647$ раз. Итак, для того чтобы произвести поворот вектора на некоторый угол, необходимо представить данный угол в виде последовательности управляющих операторов σ_i. Для нахождения данной последовательности может быть использована как табличная реализация, так и способ, когда операторы определяются, исходя из рекуррентных соотношений. Метод определения зависит от требуемого результата и приведен ниже.

Алгоритм CORDIC может быть использован в двух режимах — «поворот» и «вектор». В первом осуществляется произвольный поворот на плоскости, во втором вектор стягивается к одной из координатных осей, что используется для нахождения длины вектора. В режиме операции «поворот» управляющие операторы выбираются по формулам

$$\begin{aligned} \sigma_i &= sign(z_i) \\ z_{i+1} &= z_i - \sigma_i \alpha_i \end{aligned} \tag{5}$$

$$i = \overline{0, n-1}, \quad z_0 = \phi, \quad \alpha_i = arctg(2^{-i}).$$

Таким образом, величина $z_i \to 0$. Второй вариант — выбор управляющей последовательности таким образом, что $y_i \to 0$. В результате вектор стягивается к оси x. Эта операция называется операцией «вектор» и определяется следующими соотношениями:

$$\begin{aligned} &\sigma_i = -sign(y_i) \\ &z_{i+1} = z_i - \sigma_i \alpha_i \\ &i = \overline{0, n-1}, \quad z_0 = 0, \quad \alpha_i = arctg(2^{-i}). \end{aligned} \tag{6}$$

После выполнения n итераций получим

$$x_n \approx K_n\sqrt{x_0^2+y_0^2}, \quad y_n \approx 0, \quad z_n \approx arctg(\frac{y_0}{x_0}).$$

То есть одновременно можно вычислить как длину вектора, так и его угол относительно оси x. Величина деформации K_n — та же самая, что и для операции «поворот». Таким образом, варьируя начальные значения и используя различные правила для определения значений управляющих операторов, можно вычислять основные тригонометрические функции, а также осуществлять преобразования координат.

3.1. Вычисление синуса и косинуса. Выбирая в качестве начальных данных $x_0 = 1$, $y_0 = 0$, $z_0 = \phi$, после выполнения n итераций в режиме операции «поворот», получим в результате значения: $x_n = K_n \cos\phi$, $y_n = K_n \sin\phi$. Для получения точных значений необходимо масштабирование — умножение на величину, обратную K_n. Другой вариант реализации заключается в подстановке в итерационные соотношения в качестве начального условия $x_0 = K_n^{-1}$.

3.2. Переход от полярных координат к декартовым. Данное преобразование осуществляется по формулам $x = \rho\cos\phi$, $y = \rho\sin\phi$, для чего достаточно положить $x_0 = \rho K_n^{-1}$, $y_0 = 0$, $z_0 = \phi$.

3.3. Вычисление арктангенса. Данная функция может быть вычислена в режиме операции «вектор», для чего достаточно представить аргумент в виде частного y_0/x_0, где x_0, y_0 — нормализованные двоичные дроби. Подставляя эти значения в качестве начальных и принимая $z_0 = 0$, после выполнения n итераций получим $z_n = arctg\frac{y_0}{x_0}$.

3.4. Вычисление длины вектора. После выполнения n итераций в режиме «вектор» с взятыми в качестве начальных условий координатами вектора и $z_0 = 0$, получим $x_n = K_n\sqrt{x_0^2+y_0^2}$. Данный результат также должен быть масштабирован.

3.5. Вычисление обратных функций. Для нахождения обратных тригонометрических функций, таких, как арксинус и арккосинус, может быть использован режим операции «поворот» с некоторыми модификациями функции выбора операторов [1, 20].

Однако, в зависимости от величины аргумента, на некоторых итерациях возможен неверный выбор направления поворота. Для устранения этого недостатка предложен метод кратных итераций, который заключается в повторении некоторых итераций. Принцип использования фиксированных заранее приращений позволил расширить алгоритмы CORDIC для вычисления гиперболических функций и операций умножения–деления.

3.6. Операции умножения и деления. Определим следующие рекуррентные соотношения

$$\begin{aligned} x_{i+1} &= x_i - 0 \cdot y_i \sigma_i 2^{-i} = x_i \\ y_{i+1} &= y_i + x_i \sigma_i 2^{-i} \\ z_{i+1} &= z_i - \sigma_i 2^{-i} \end{aligned} \tag{7}$$

Правила выбора управляющих операторов остаются такими же, как и в случае исходного CORDIC-алгоритма (5), (6). Ниже приводятся результаты вычислений в режиме операции «поворот» и в режиме операции «вектор» после выполнения n итераций.

$$x_n = x_0, y_n = y_0 + x_0 z_0, z_n = 0. \tag{8}$$

$$x_n = x_0, y_n = 0, z_n = z_0 - \frac{y_0}{x_0}. \tag{9}$$

Варьируя начальные значения, можно получить несколько операций множительно-делительного типа (к примеру, умножение с накоплением).

Как можно легко заметить, все рекуррентные соотношения в алгоритмах CORDIC являются однотипными с некоторыми модификациями для каждого конкретного случая. В работе [29] была впервые предложена некоторая обобщенная форма записи всех возможных модификаций алгоритма.

$$\begin{aligned} x_{i+1} &= x_i - m y_i \sigma_i 2^{-i} \\ y_{i+1} &= y_i + x_i \sigma_i 2^{-i} \\ z_{i+1} &= z_i - \sigma_i \alpha_i \end{aligned} \tag{10}$$

Здесь параметр m определяет систему, в которой действует алгоритм: $m = 1$ — обычный поворот вектора на плоскости, $m = 0$ —

множительно-делительные операции (линейная система), $m = -1$ — поворот в гиперболических координатах. Вспомогательные величины α_iзависят от выбранной системы и определяются следующим образом: $\alpha_i = arctg 2^{-i}$ при $m = 1$, $\alpha_i = 2^{-i}$ при $m = 0$, $\alpha_i = arcth 2^{-i}$ при $m = -1$. Правила выбора для управляющей последовательности операторов σ_i зависят от выбранного режима операций и определяются формулами (5) либо (6) соответственно.

4. Методы коррекции результата и ускорения вычислений

4.1. Компенсация коэффициента деформации. Одним из недостатков, ограничивающих алгоритмы CORDIC по скорости выполнения, является необходимость коррекции результата. Результаты x_n, y_n необходимо умножить на величину, обратную коэффициенту деформации K, для компенсации деформации вектора в процессе выполнений итераций. Однако, выполнение операции умножения ведет к существенной задержке в вычислениях. Уменьшению этой задержки посвящен ряд работ. Ниже приводятся два алгоритма, предложенных для компенсации постоянного коэффициента деформации.

4.1.1. *Метод двойного поворота.* Введем в рассмотрение угол

$$\beta = arccos(\frac{1}{K}) .$$

Метод двойного поворота [26] заключается в том, что операция поворота на угол ϕ производится, как две отдельные операции «поворот» на углы $(\phi - \beta)$ и $(\phi + \beta)$. Производя простейшие тригонометрические преобразования, получим

$$\frac{x^+ + x^-}{2} = K \cos\beta(x_0 \cos\phi - y_0 \sin\phi)$$
$$\frac{y^+ + y^-}{2} = K \cos\beta(x_0 \sin\phi + y_0 \cos\phi)$$

Правая часть формул является результатом, не требующим коррекции, операции «поворот» в силу определения угла β. Данный способ легко реализуем, однако требует дублирования аппаратуры для реализации двух параллельно исполняемых операций «поворот». Кроме того, он работает только тогда, когда заранее известен коэффициент деформации и не годится для тех CORDIC-схем, в которых этот коэффициент зависит от величины угла поворота. Разновидность данного метода для операции «вектор» для случая, когда угол положения вектора известен, рассмотрена также в работе [33].

4.1.2. *Метод анализа битов переноса.* Данный метод основан на анализе отдельных битов величин x_i, y_i в процессе итераций. В результате появляется возможность вычисления отдельно точных значений *xcomp*, *ycomp*. На каждой итерации, начиная с номера $i = 2$, вычисляются значения

$$\delta_j = x_{-j}(i) + 2(c_{-j} - y_{2*})$$
$$\mu_j = y_{-j}(i) + 2(c'_{-j} - x_{2*}).$$

Здесь $x_{-j}(i)$, $y_{-j}(i)$ — двоичные разряды величин x_i, y_i с весом 2^{-j}, c_{-j}, c'_{-j} — возможный перенос (заем) из j-й позиции при суммировании или вычитании, x_{2*}, y_{2*} — равняются нулю в случае избыточного представления цифр со знаком (signed-digit representation) или соответствуют знаковым битам величин $\sigma_i 2^{-i} x_i$ и $\sigma_i 2^{-i} y_i$ в случае обычного двоичного представления. Для получения точных значений используются итерации вида

$$xcomp_j = xcomp_{j-1} + \delta_j 2^{-j} K^{-1}$$
$$ycomp_j = ycomp_{j-1} + \mu_j 2^{-j} K^{-1}$$

Таким образом, компенсация коэффициента деформации может проводиться параллельно стандартным итерациям алгоритма путем последовательно выполняемых операций сложения и сдвига. Данный алгоритм может применяться как в режиме операции «поворот», так и в режиме операции «вектор», то есть является универсальным, однако, также требует знания коэффициента деформации заранее.

4.2. Гибридные схемы. Для сокращения числа итераций используются алгоритмы с большим основанием 8 и техники избыточного сложения. Их применение влечет за собой необходимость параллельного вычисления коэффициента деформации, либо введения специальных корректирующих итераций. В любом случае общее число итераций увеличивается. Одним из возможных путей получения алгоритмов максимального быстродействия с минимальным числом итераций и постоянным коэффициентом деформации является комбинированный подход. К этому же классу схем могут быть отнесены алгоритмы со смешанным основанием [5]. Ниже приводится описание способа, предложенного в работе [7] для операции «поворот».

Методика предсказания направления поворота основана на том, что значения операторов выбираются в соответствии с цифрами величины z, представленной в системе счисления с симметричной базой: $z_j = \sum_{i=j}^{n} c_j 2^{-j} = \sum_{i=j}^{n} \sigma_j \alpha_j$, где $\sigma_j, c_j \in \{-1, +1\}$, $\alpha_j = arctg 2^{-j}$. Отметим, что при таком определении операторов коэффициент не изменяется. Так как, вообще говоря, $\alpha_j \neq 2^{-j}$, для обеспечения сходимости вводятся специальные корректирующие итерации. При выборе значений операторов $\sigma_i = c_i, i = j \dots k$ для коррекции накопленной ошибки необходимо повторить итерацию с номером k. Индексы j и k должны удовлетворять условию исправляемости итераций: $\sum_{i=j}^{k} \left|2^{-i} - \alpha_i\right| \leq 2^{-k} \rightarrow k \leq 3j + 1$. Таким образом, итерации с номерами $k = 1, 4, 13, \dots$ должны быть повторены. Недостатком данного подхода является необходимость перехода от избыточного представления чисел к системе счисления с симметричной базой перед каждым предсказанием последующих значений операторов.

В работе [7] предлагается следующая модификация алгоритма. Для того чтобы сохранить постоянный коэффициент деформации, достаточно, чтобы $\sigma_i \in \{-1, +1\}$ для $i < \frac{n}{4}$. Для итераций с номерами $i \geq \frac{n}{4}$ можно использовать избыточное множество значений операторов, включающее 0. Для этой цели всякий раз, когда выбирается $\sigma_i = 0$, производится масштабирование вектора на величину $1 + 2^{-2i-1}$. Предлагаемый алгоритм носит гибридный характер.

На итерациях с номерами от 0 до $\frac{n}{4}$ используется методика предсказания направления поворота со значениями $\sigma_i \in \{-1, +1\}$. Так как условие не выполняется, то используются микроповороты с множественными сдвигами $\alpha_i = arctg(2^{-i} + s_{i1} 2^{-d1} + s_{i2} 2^{-d2} + \dots s_{ik} 2^{-dk})$, где $s_{ij} \in \{-1, 0, +1\}$, $i < d_{ij} < n$, аналогично [32]. Данная замена используется только на первых нескольких итерациях, а соответствующие величины s_{ij}, d_{ij} находятся предварительно. Так как теперь операторы поворота доступны одновременно, величина $z_{\frac{n}{4}+1}$ может быть быстро вычислена с помощью каскада сумматоров с сохранением переноса (carry-save adders). Затем данная величина преобразуется к избыточной системе цифр со знаком и используется для предсказания значений операторов с номерами $i \geq \frac{n}{4}$ выбор $\sigma_i = 0$ не влечет за собой никаких осложнений. Для итераций с номерами $i \geq \frac{n}{2}$ используется алгоритм по основанию 4 с целью сокращения общего числа итераций.

4.3. Особенности аппаратной реализации. Наиболее простой способ реализации алгоритмов CORDIC заключается в обычном дублировании трех соответствующих рекуррентных соотношений в аппаратуре. Структура развернута в пространстве в виде трех столбцов сумматоров и устройств определения направления поворота. Преимущество данной структуры перед соответствующей итеративной реализацией — в возможности зашить в аппаратуру фиксированные заранее для каждой стадии сдвиги и константы (сдвиг на переменное число разрядов не является быстрой операцией). Для достижения большей пропускной способности всей системы в целом рекомендована конвейеризованная архитектура, отличающаяся наличием регистров между стадиями для сохранения промежуточных результатов [30, 31].

Наиболее часто используемой операцией является сложение. Поэтому для достижения наибольшего быстродействия были испробованы всевозможные конструкции сумматоров [27]. Основным недостатком традиционных устройств сложения–вычитания является то, что время суммирования зависит от наличия переноса от младших разрядов к старшим и определяется как $O(\log n)$, где n — число разрядов. Для устранения этого недостатка были предложены так называемые избыточные сумматоры (redundant adders), свободные от распространения переноса. Суммирование в таких устройствах занимает небольшое время, не зависящее от длины разрядной сетки. Недостатком избыточных сумматоров является то, что они требуют вдвое больше аппаратных затрат.

Необходимым условием для применения быстрого сложения является то, что операнды также должны быть представлены в избыточной форме. Обычно используются два таких представления — представление с сохранением переноса (carry–save representation) или представление в виде цифр со знаком (signed–digit representation). В зависимости от выбранного способа представления изменяется как аппаратная реализация алгоритмов, так и сами алгоритмы.

Данные техники позволяют существенно сократить время выполнения операции определения знака, но попутно влекут за собой следующую проблему — при данном способе определения направления поворотов операторы $\sigma_i \in \{-1, 0, +1\}$, следовательно, коэффициент деформации $K = \prod_{i=0}^{n-1} \sqrt{1 + \sigma_i^2 2^{-2i}}$ уже не является константой. Следовательно, необходима дополнительная аппаратура для вычисления

коэффициента деформации в соответствии с выбранной последовательностью операторов. Типичный подход в этом случае — комбинация табличного и алгоритмического методов. В соответствии с требуемой точностью после нескольких первых итераций значение коэффициента деформации выбирается из таблицы (look-up le), которое затем корректируется при помощи последовательности сложений и сдвигов.

Второй вариант — попытаться сохранить прежнее множество значений операторов с целью сохранить постоянный коэффициент деформации. В этом случае на некоторых итерациях возможен неправильный выбор управляющих операторов, нарушающий условия сходимости алгоритма. Для восстановления условий сходимости требуется повторение некоторых итераций.

Еще один способ, разработанный с целью преодоления данного недостатка, — алгоритм с ветвлением (branching Cordic algorithm), который позволяет сохранить неизбыточное множество значений операторов, не прибегая к дополнительным итерациям [13, 22]. В случае, когда по оценке нельзя с уверенностью определить знак числа из-за большого количества нулей в старших разрядах, данный алгоритм разветвляется. Одна ветка вычислений предполагает $\sigma_i = +1$, другая $\sigma_i = -1$. На основе сравнительного анализа результатов выбирается та ветка, исполнение которой не противоречит условиям сходимости. Однако в этом случае также требуется дополнительная аппаратура.

5. Разрядно-параллельные CORDIC-алгоритмы

Исследования принципиальной возможности представления алгоритмов CORDIC в виде разрядных соотношений между битами аргумента и результата были выполнены в работах [33–35]. Для перехода от итерационных соотношений к параллельным использовался разрядный подход, совмещенный с последовательным раскрытием циклов. В дальнейшем изложении мы несколько изменим традиционную запись соотношений, что ни в коем случае не меняет сущности алгоритма. В новом варианте записи рекуррентные соотношения

Волдера для вычисления новых координат X', Y' при повороте вектора на угол φ имеют следующий вид:

$$(11)\qquad \begin{aligned} \varphi_{i+1} &= \varphi_i - \sigma_i arctg(2^{-i}) \\ \sigma_{i+1} &= sign(\varphi_{i+1}) \\ y_{i+1} &= y_i - \sigma_i x_i \cdot 2^{-i} \\ x_{i+1} &= x_i + \sigma_i y_i \cdot 2^{-i} \end{aligned}$$

Начальные значения: $\varphi_0 = \varphi$, $\sigma_0 = \pm 1$ (знак выбирается в зависимости от направления вращения вектора), $x_0 = X, \quad y_0 = Y$. Результат:

$$y_n = Y' = K_n[Y\cos(\varphi) + X\sin(\varphi)],\ x_n = X' = K_n[X\cos(\varphi) - Y\sin(\varphi)].$$

Для разрядности $n \geq 10$ имеем $K_n = K \approx 1,65$.

Введем обозначения: $A = Y - \sigma_0 X$, $B = X + \sigma_0 Y$. Здесь X, Y — начальные координаты конца вектора, причем $X = X_1X_2\ldots X_8$, $Y = Y_1Y_2\ldots Y_8$, $A = a_1a_2\ldots a_8$, $B = b_1b_2\ldots b_8$ — разрядные двоичные представления. Пусть координаты конца повернутого вектора есть: $X' = X'_1X'_2\ldots X'_8$, $Y' = Y'_1Y'_2\ldots Y'_8$. Тогда, на основании работ [33–35], для координат X', X' можно записать разрядно-параллельные схемы вычисления новых координат. Соответствующие результаты для $n = 10$ представлены в Таблице 1. Здесь переменными являются величины $A(X,Y) = Y - \sigma_0 X$ и $B(X,Y) = X + \sigma_0 Y$, производные которых равны, соответственно, $\frac{\partial A}{\partial X} = -\sigma_0$, $\frac{\partial B}{\partial X} = 1$. С учетом этого запишем выражение для полной производной одной из новых координат:

$$(12)\qquad \frac{\partial x_{10}}{\partial X} = \frac{\partial x_{10}}{\partial A}\cdot\frac{\partial A}{\partial X} + \frac{\partial x_{10}}{\partial B}\cdot\frac{\partial B}{\partial X} = \frac{\partial x_{10}}{\partial A} + \sigma_0\frac{\partial x_{10}}{\partial B}$$

С другой стороны,

$$\frac{\partial x_{10}}{\partial X} = K_n\cos\varphi.$$

Очевидно, подставляя в формулы (12) результат дифференцирования разрядных схем Таблицы 1, мы получим новые схемы для параллельного вычисления функции $\cos(\varphi)$ при условии соответствующей коррекции. Аналогично для вычисления функции $\sin(\varphi)$следует взять производную $\frac{\partial x_{10}}{\partial Y}$, учитывая, что $\frac{\partial A}{\partial Y} = 1$, $\frac{\partial B}{\partial Y} = \sigma_0$. Полностью результаты приведены в Таблице 2. Запишем далее выражения для полных производных.

$$(13)\qquad \begin{aligned} \tfrac{\partial x_{10}}{\partial X} + \tfrac{\partial x_{10}}{\partial Y} &= K(\cos(\varphi) - \sin(\varphi)) = \tfrac{\partial y_{10}}{\partial X} - \tfrac{\partial y_{10}}{\partial Y} \\ \tfrac{\partial y_{10}}{\partial X} + \tfrac{\partial y_{10}}{\partial Y} &= K(\cos(\varphi) + \sin(\varphi)) = \tfrac{\partial x_{10}}{\partial X} - \tfrac{\partial x_{10}}{\partial Y} \end{aligned}$$

Вес	x_{10}	y_{10}
2^0	B	A
2^{-1}	$\sigma_1 A$	$-\sigma_1 B$
2^{-2}	$\sigma_2 A$	$-\sigma_2 B$
2^{-3}	$\sigma_3 A - \sigma_1\sigma_2 B$	$-\sigma_3 B - \sigma_1\sigma_2 A$
2^{-4}	$\sigma_4 A - \sigma_1\sigma_3 B$	$-\sigma_4 B - \sigma_1\sigma_3 A$
2^{-5}	$\sigma_5 A - \sigma_1\sigma_4 B - \sigma_2\sigma_3 B$	$-\sigma_5 B - \sigma_1\sigma_4 A - \sigma_2\sigma_3 A$
2^{-6}	$\sigma_6 A - \sigma_1\sigma_5 B - \sigma_2\sigma_4 B -$ $-\sigma_1\sigma_2\sigma_3 A$	$-\sigma_6 B - \sigma_1\sigma_5 A - \sigma_2\sigma_4 A +$ $+\sigma_1\sigma_2\sigma_3 B$
2^{-7}	$\sigma_7 A - \sigma_1\sigma_6 B - \sigma_2\sigma_5 B -$ $-\sigma_3\sigma_4 B - \sigma_1\sigma_2\sigma_4 A$	$-\sigma_7 B - \sigma_1\sigma_6 A - \sigma_2\sigma_5 A -$ $-\sigma_3\sigma_4 A + \sigma_1\sigma_2\sigma_4 B$
2^{-8}	$\sigma_8 A - \sigma_1\sigma_7 B - \sigma_2\sigma_6 B -$ $-\sigma_3\sigma_5 B - \sigma_1\sigma_2\sigma_5 A -$ $-\sigma_1\sigma_3\sigma_4 A$	$-\sigma_8 A - \sigma_1\sigma_7 A - \sigma_2\sigma_6 A -$ $-\sigma_3\sigma_5 A + \sigma_1\sigma_2\sigma_5 B +$ $+\sigma_1\sigma_3\sigma_4 B$
2^{-9}	$\sigma_9 A - \sigma_1\sigma_8 B - \sigma_2\sigma_7 B -$ $-\sigma_3\sigma_6 B - \sigma_4\sigma_5 B -$ $-\sigma_1\sigma_2\sigma_6 A - \sigma_1\sigma_3\sigma_5 A -$ $-\sigma_2\sigma_3\sigma_4 A$	$-\sigma_9 B - \sigma_1\sigma_8 A - \sigma_2\sigma_7 A -$ $-\sigma_3\sigma_6 A - \sigma_4\sigma_5 A +$ $+\sigma_1\sigma_2\sigma_6 B + \sigma_1\sigma_3\sigma_5 B +$ $+\sigma_2\sigma_3\sigma_4 B$

Таблица 1. Разрядно-параллельные формулы поворота

Для вычисления указанных функций достаточно произвести поразрядное суммирование или вычитание схем Таблицы 2. Для параллельного определения операторов Волдера воспользуемся соотношением $\lim\limits_{n\to\infty}(\varphi - \sum\limits_{i=0}^{n}\sigma_i arctg(2^{-i}) = 0$, откуда при ограниченном значении n следует

$$\varphi = \sum_{i=0}^{\infty}\sigma_i arctg(2^{-i}) \approx \sum_{i=0}^{n}\sigma_i arctg(2^{-i}) \tag{14}$$

Величины $arctg(2^{-i})$ являются табличными константами, см. Таблицу 3.

Умножая на σ_i двоичные представления $arctg(2^{-i})$ и суммируя в соответствии с (14) получим разрядное представление угла поворота

Вес	$\frac{\partial x_{10}}{\partial X} = K_n \cos\varphi$	$\frac{\partial x_{10}}{\partial Y} = K_n \sin\varphi$
2^0	1	σ_0
2^{-1}	$-\sigma_0\sigma_1$	σ_1
2^{-2}	$-\sigma_0\sigma_2$	σ_2
2^{-3}	$-\sigma_0\sigma_3 - \sigma_1\sigma_2$	$\sigma_3 - \sigma_0\sigma_1\sigma_2$
2^{-4}	$-\sigma_0\sigma_4 - \sigma_1\sigma_3$	$\sigma_4 - \sigma_0\sigma_1\sigma_3$
2^{-5}	$-\sigma_0\sigma_5 - \sigma_1\sigma_4 - \sigma_2\sigma_3$	$\sigma_5 - \sigma_0\sigma_1\sigma_4 - \sigma_0\sigma_2\sigma_3$
2^{-6}	$-\sigma_0\sigma_6 - \sigma_1\sigma_5 - \sigma_2\sigma_4 + +\sigma_0\sigma_1\sigma_2\sigma_3$	$\sigma_6 - \sigma_0\sigma_1\sigma_5 - \sigma_0\sigma_2\sigma_4 - -\sigma_1\sigma_2\sigma_3$
2^{-7}	$-\sigma_0\sigma_7 - \sigma_1\sigma_6 - \sigma_2\sigma_5 - -\sigma_3\sigma_4 + \sigma_0\sigma_1\sigma_2\sigma_4$	$\sigma_7 - \sigma_0\sigma_1\sigma_6 - \sigma_0\sigma_2\sigma_5 - -\sigma_0\sigma_3\sigma_4 - \sigma_1\sigma_2\sigma_4$
2^{-8}	$-\sigma_0\sigma_8 - \sigma_1\sigma_7 - \sigma_2\sigma_6 - -\sigma_3\sigma_5 + \sigma_0\sigma_1\sigma_2\sigma_5 + +\sigma_0\sigma_1\sigma_3\sigma_4$	$\sigma_8 - \sigma_0\sigma_1\sigma_7 - \sigma_0\sigma_2\sigma_6 - -\sigma_0\sigma_3\sigma_5 - \sigma_1\sigma_2\sigma_5 - -\sigma_1\sigma_3\sigma_4$
2^{-9}	$-\sigma_0\sigma_9 - \sigma_1\sigma_8 - \sigma_2\sigma_7 - -\sigma_3\sigma_6 + \sigma_0\sigma_1\sigma_2\sigma_6 + +\sigma_0\sigma_1\sigma_3\sigma_5 + \sigma_0\sigma_2\sigma_3\sigma_4$	$\sigma_9 - \sigma_0\sigma_1\sigma_8 - \sigma_0\sigma_2\sigma_7 - -\sigma_0\sigma_3\sigma_6 - \sigma_0\sigma_4\sigma_5 - -\sigma_1\sigma_2\sigma_6 - \sigma_1\sigma_3\sigma_5 - \sigma_2\sigma_3\sigma_4$

Таблица 2. Производные для формул поворота

i	$arctg(2^{-i})$град (decimal)	$arctg(2^{-i})$град (binary)
0	45.0	101101.00
1	26.6	011010.10
2	14.0	001110.00
3	7.2	000111.00
4	3.6	000011.10
5	1.8	000001.11
6	0.9	000000.11
7	0.45	000000.01

Таблица 3. Значения арктангенса

$\varphi = (\varphi_1\varphi_2\ldots\varphi_n)$:

$$
\begin{aligned}
\varphi_1 &= \sigma_0 \\
\varphi_2 &= \sigma_1 \\
\varphi_3 &= \sigma_0 + \sigma_1 + \sigma_2 \\
\varphi_4 &= \sigma_0 + \sigma_2 + \sigma_3 \\
\varphi_5 &= \sigma_1 + \sigma_2 + \sigma_3 + \sigma_4 \\
\varphi_6 &= \sigma_2 + \sigma_3 + \sigma_4 + \sigma_5
\end{aligned} \tag{15}
$$

Используя данные представления, получим псевдооператоры $\tilde{\sigma}_i$, которые отличаются от операторов σ_i тем, что они, в общем случае, не отвечают условию $\sigma_i \in \{+1, -1\}$, $\quad i = 1 \dots n$:

$$
(16) \qquad \begin{aligned}
\tilde{\sigma}_0 &= \varphi_1 \\
\tilde{\sigma}_1 &= \varphi_2 \\
\tilde{\sigma}_2 &= \varphi_3 - \varphi_1 - \varphi_2 \\
\tilde{\sigma}_3 &= \varphi_4 + \varphi_2 - \varphi_3 \\
\tilde{\sigma}_4 &= \varphi_5 + \varphi_1 - \varphi_4 - \varphi_2 \\
\tilde{\sigma}_5 &= \varphi_6 - 2\varphi_1 + \varphi_3 - \varphi_5 \\
\tilde{\sigma}_6 &= \varphi_7 + \varphi_4 - \varphi_6 - \varphi_3 \\
\tilde{\sigma}_7 &= \varphi_8 + 2\varphi_1 + \varphi_5 - \varphi_7 - \varphi_4
\end{aligned}
$$

С учетом одновременной доступности $\tilde{\sigma}_i$ схемы Таблицы 2 следует считать параллельными. Окончательное представление можно получить путем непосредственной подстановки $\tilde{\sigma}_i$ (16) вместо операторов σ_i [35].

Погрешность вычислений, полученная в результате компьютерного моделирования разрядных схем представлена на Рисунке 1. Для реализации операции «вектор» достаточно приравнять нулю выражения для всех разрядов Y', что равносильно повороту вектора до его совмещения с осью абсцисс. Полученная при этом система псевдооператоров должна быть подставлена в выражения для вычисления длины вектора X' и в выражение (14) для вычисления угла положения.

6. Реализация трехмерных CORDIC операций

Двумерные операции «вектор» и «поворот» можно использовать для определения длины пространственного вектора [1, 36]. Так например, величину $d = \sqrt{X^2 + Y^2 + Z^2}$ можно вычислить, дважды применяя стандартный итерационный процесс «вектор»:

1-ый этап:

$$
\begin{cases}
y_{i+1} &= y_i - \sigma_{1,i} \cdot x_i \cdot 2^{-i} \\
x_{i+1} &= x_i + \sigma_{1,i} \cdot y_i \cdot 2^{-i} \\
\sigma_{1,i+1} &= sign(y_{i+1})
\end{cases}
$$

Начальные условия: $x_0 = X$, $y_0 = Y$, $\sigma_{1,0} = 1$.

Результат: $x_n = K_n\sqrt{X^2 + Y^2}$, $y_n = 0$.

2-ой этап:

$$
\begin{cases}
x_{i+1} &= x_i - \sigma_{2,i} \cdot z_i \cdot 2^{-i} \\
z_{i+1} &= z_i + \sigma_{2,i} \cdot x_i \cdot 2^{-i} \\
\sigma_{2,i+1} &= sign(x_{i+1})
\end{cases}
$$

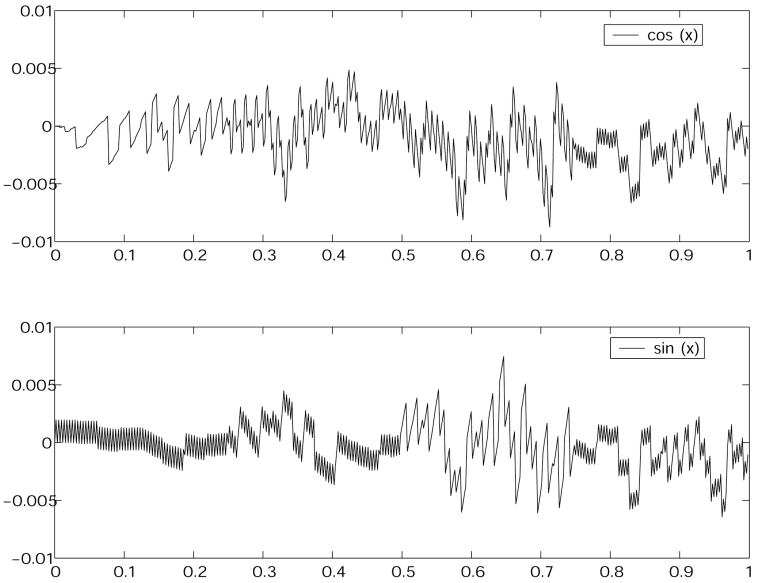

Рис. 1. Погрешность вычислений по разрядным схемам для тригонометричесих функций

Начальные условия: $x_0 = x_n/K_n$, $z_0 = Z$, $\sigma_{2,0} = 1$.
Результат: $z_n = K_n\sqrt{X^2 + Y^2 + Z^2}$, $x_n = 0$.

Причем коррекция результата первого этапа, связанная с устранением деформации вектора, является обязательной до применения второго этапа. Деформация второго этапа устраняется перечисленными выше методами коррекции.

Представляет теоретический интерес разработка одноэтапной процедуры измерения длины пространственного вектора. Итерационные формулы в этом случае приобретают следующий вид:

$$(17)\qquad \begin{cases} \begin{cases} y_{i+1} &= y_i - \sigma_{1,i}\cdot x_i\cdot 2^{-i} \\ z_{i+1} &= (z_i + \sigma_{1,i}\cdot y_i\cdot 2^{-i})\cdot {K_i}^{-1} \\ \sigma_{1,i+1} &= sign(y_{i+1}) \end{cases} & z_i = z_{i+1} \\ \begin{cases} x_{i+1} &= x_i - \sigma_{2,i}\cdot z_i\cdot 2^{-i} \\ z_{i+1} &= z_i + \sigma_{2,i}\cdot x_i\cdot 2^{-i} \\ \sigma_{2,i+1} &= sign(x_{i+1}) \end{cases} \end{cases}$$

Начальные значения: $x_0 = X$, $y_0 = Y$, $z_0 = Z$, $\sigma_{1,0} = \sigma_{2,0} = 1$.
Результат: $z_n = K_n\sqrt{X^2 + Y^2 + Z^2}$, $x_n = 0$, $y_n = 0$.
Здесь коррекция введена только для одной итерационной формулы. Величины $K_i = \sqrt{1 + 2^{-2i}}$ вычисляются заранее.
Исследования показали, что другие варианты отличаются либо разбросом величины коэффициента деформации вектора

$$K_n \in (1.63, 1.66)$$

и могут быть использованы только для грубых измерений, либо большими аппаратурными затратами.
Вычислительная схема (17) является наиболее компактной и потому более удобной для распараллеливания. На ее основе можно получить и разрядно-параллельные представления, пользуясь рассмотренным ранее приемом раскрытия итераций.

7. Заключение

В настоящей работе систематизированы последние результаты в области алгоритмов CORDIC. Показано, что к данному направлению не ослабевает интерес, в связи с большими перспективами использования. Алгоритмы семейства CORDIC обладают универсальностью, их целесообразно использовать для аппаратной реализации элементарных функций, цифровых фильтров, алгоритмов машинной графики и др. Приведена классификация алгоритмов CORDIC с указанием работ последних лет. Рассмотрены методы компенсации деформации вектора. Предложен метод получения разрядно-параллельных форм, обеспечивающий максимальное распараллеливание итерационных алгоритмов семейства CORDIC и расширение функциональных возможностей. Данное представление алгоритмов может оказаться эффективным при проектировании алгоритмического обеспечения специализированных вычислителей на основе программируемых логических матриц. Рассмотрены различные способы получения трехмерных операций «вектор», расширяющие возможности CORDIC-алгоритмов.

Список литературы

[1] Байков В. Д., Смолов В. Б. Аппаратурная реализация элементарных функций в ЦВМ. — Ленинград: Издательство ЛГУ, 1975. ↑2, 2, 3.5, 6

[2] Пухов Г. Е., Евдокимов В. Ф., Синьков М.В. Разрядно-аналоговые вычислительные системы. — Москва: Сов. Радио, 1978. ↑

[3] Andraka R. *A survey of Cordic algorithms for FPGA based computers* // ACM/SIGDA 6^{th} International Symposium on FPGAs, 1998, с. 1981–2000. ↑2, 2

[4] Antelo E., Bruguera J. D., Lang T., Zapata E. L. *Error analysis and reduction for angle calculation using the Cordic algorithm*: Internal Report Dept. of Electrical and Comp. Eng., UCI, 1996. ↑2

[5] Antelo E., Bruguera J. D., Zapata E. L. *Unified mixed radix 2-4 redundant Cordic rocessor* IEEE Transactions on Computers. — Т. **45, no.9**, 1996, с. 1068–1073. ↑2, 4.2

[6] Antelo E., Bruguera J. D., Lang T., Villalba J., Zapata E. L. *High perfomance rotation architectures based on the radix-4 Cordic algorithm* IEEE Transactions on Computers. — Т. **46, no.8**, 1997, с. 855–870. ↑2

[7] Antelo E., Bruguera J. D., Villalba J., Zapata E.L. *Redundant Cordic rotator based on parallel prediction* // 12^{th} Symposium on Computer Arithmetic, 1995, с. 172–179. ↑2, 4.2

[8] Antelo E., Lang T., Bruguera J.D. Very-high radix Cordic rotation based on selection by rounding, 2000. ↑2

[9] Bruguera J. D., Guil N., Lang T., Villalba J., Zapata E. L. Cordic based parallel/pipelined architecture for the Hough transform, 1996, с. 1–18. ↑2, 3

[10] Cavallaro J. R., Luk F. T. Cordic arithmetic for a SVD processor. — Т. **5**, 1988, с. 271–290. ↑2

[11] Despain A. M *Fourier transform computers using Cordic iterations* IEEE Transactions on Computers. — Т. **C-30**, 1974, с. 993–1001. ↑1

[12] Douknitch E. I. *Highly parallel multidimensional Cordic-like algorithms* // Int. Conf. Intelligent Multiprocessor Systems, 1999, с. 44–48. ↑2

[13] Duprat J., Muller J. M. *The Cordic algorithm: new results for fast VLSI implementation* IEEE Transactions on Computers. — Т. **42**, 1993, с. 168–178. ↑2, 4.3

[14] Ercegovac M. D., Lang T. *Redundant and on-line Cordic: application to matrix triangularization and SVD* IEEE Transactions on Computers. — Т. **39, no.6**, 1990, с. 725–740. ↑2, 2

[15] Hekstra G. J., Deprettere F. A. *Floating point Cordic* // 11^{th} Synposium on Computer Arithmetic, 1993. ↑2

[16] Hsiao S., Delosme J.–M. *The Cordic Householder algorithm*. — 10^{th} Symposium on Computer Arithmetic, 1991, с. 256–263. ↑2

[17] Hu Y., Naganathan S. *An angle recoding method for Cordic algorithm implementation* IEEE Transactions on Computers. — Т. **42**, 1993, с. 99–102. ↑2

[18] Hu Y. Cordic-based VLSI architectures for digital signal processing, 1992, с. 6–35. ↑2, 3

[19] Hu Y. *The quantization effects on the Cordic algorithm* IEEE Transactions on Computers. — T. **40, no.4**, 1992, c. 834–844. ↑2

[20] Lang T., Antelo E. *Cordic vectoring with arbitrary target value* // 13^{th} IEEE Symposium on Computer Arithmetic, 1997, c. 108–115. ↑2, 3.5

[21] Osorio R., Antelo E., Bruguera J. D., Villalba J., Zapata E. L. *Digit online large radix Cordic rotator* // Conf. on Application-Specific Array Processors IEEE, 1995. ↑2

[22] Phatak D. S. *Double step branching Cordic: a new algorithm for fast sine and cosine generation* IEEE Transactions on Computers. — T. **47, no.5**, 1998, c. 587–602. ↑2, 4.3

[23] Timmermann D., Hann H., Hostika B. J. *Low latency time Cordic algorithms* IEEE Transactions on Computers. — T. **41**, 1992, c. 1010–1015. ↑2

[24] Valls J., Kuhlmann M., Parhi K. *Efficient mapping of Cordic algorithm on FPGA* // IEEE Workshop on Signal Processing Systems, 2000. ↑2

[25] Villalba J., Lang T. *Low latency word serial Cordic* // Application-Specific Systems, Architectures and Processors Conf./ IEEE, 1997, c. 124–131. ↑

[26] Villalba J., Lang T., Zapata E. L. Parallel compensation of scale factor for the Cordic algorithm. — T. **19, no. 3**, 1998, c. 227–241. ↑2, 4.1.1

[27] Vladimirova T., Tiggler H. *FPGA implementation of sine and cosine generators using Cordic algorithm* // Military and Aerospace Applications of Programmable Devices and Technologies Conference, 1998, c. 28–30. ↑2, 4.3

[28] Volder J. E. *The Cordic trigonometric computing technique* IRE Trans. on Electronic Computers. — T. **8**, 1959, c. 330–334. ↑2, 3

[29] Walther J. S. *A unified algorithm for elementary functions* // Spring Joint Computer, 1991, c. 379–385. ↑2, 3.6

[30] Wang S., Piuri V., Swartzlander E. *Granularly-pipelined Cordic processors for sine and cosine generators*: Internal Report # 95-041, Politecnico di Milano, 1995. ↑2, 4.3

[31] Wang S., Swartzlander E. E. *Merged Cordic algorithm* // International Symp. On Circuits and Systems 1995, c. 1988–1991. ↑2, 4.3

[32] Wu C., Wu A. Y. *A novel rotational VLSI architecture based on Extended Elementary Angle Set Cordic algorithm* // 2^{th} Asia Pacific Conference on ASICs IEEE, 2000, c. 111–114. ↑2, 4.2

[33] Khachumov V. *Bit-parallel algorithms and devices* // 11^{th} International Conference on Computer graphics&vision GRAPHICON. — Nizhny Novgorod , 2001, c. 224–226. ↑4.1.1, 5, 5

[34] Захаров А. В., Хачумов В. М. *Разрядно-параллельные вычисления в системах реального времени*: Сб. трудов Математика, информатика: теория и практика. — Переславль-Залесский: Изд-во университета г.Переславля, 2003, с. 97–104. ↑

[35] Khachumov V. M. *Bit–Parallel Structures for Image Processing and Analysis* // Pattern Recognition and Image Analysis Conf. T. **13, no.4**, 2003, c. 633–639. ↑5, 5, 5

[36] Хачумов В. М. *Трехмерные операции Волдера для измерения длины вектора* // Всероссийская конф. по информационно-управляющим системам и специализированным вычислительным устройствам для обработки и передачи данных: Тезисы доклада. — Махачкала, 1996, с. 95–96. ↑6

Российский Университет Дружбы Народов

Исследовательский центр искусственного интеллекта ИПС РАН

A. V. Zakahrov, V. M. Khachumov. *Current state and prospectives of the CORDIC algorithms.* (in russian.)

Abstract. This paper presents a taxonomy of CORDIC–based algorithms which classifies the algorithms based upon their implementations. First sections of this work contains the base CORDIC theory and various extensions suggested for improving of the algorithm. Below, a new fast parallel computational approach to the CORDIC algorithm based on the bit–parallel technique is proposed. Experimental results for trigonometric functions are reported. Last section describe general 3–dimensional CORDIC algorithm.