Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра электронных вычислительных средств

М. И. Вашкевич, Н. А. Петровский, А. А. Петровский

ПРОЕКТИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ СРЕДСТВ С ДИНАМИЧЕСКИ РЕКОНФИГУРИРУЕМОЙ АРХИТЕКТУРОЙ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Рекомендовано УМО по образованию в области информатики и радиоэлектроники в качестве пособия для специальности 1-40 02 02 «Электронные вычислительные средства»

Минск БГУИР 2019

УДК 004.31-027.31(076.5)
ББК 32.973.26-04я73
В23

Рецензенты:

кафедра радиофизики и цифровых медиатехнологий
Белорусского государственного университета (протокол № 10 от 20.06.2018);

главный научный сотрудник государственного научного учреждения
«Объединенный институт проблем информатики
Национальной академии наук Беларуси»
доктор технических наук, профессор Г. И. Алексеев

Вашкевич М. И.

В23 Проектирование вычислительных средств с динамически реконфигурируемой архитектурой. Лабораторный практикум : пособие / М. И. Вашкевич, Н. А. Петровский, А. А. Петровский. – Минск : БГУИР, 2019. – 64 с. : ил.

ISBN 978-985-543-448-2.

Пособие целенаправленно показывает весь процесс проектирования реконфигурируемых CORDIC-процессоров на примере умножения кватернионов, где один из сомножителей – кватернион-константа, т. е. кватернион с постоянными параметрами. Сформулирована задача оптимизации для фиксированного угла вращения и показано ее решение для 2D и 4D CORDIC-алгоритмов. Показана схема умножения кватернионов на основе блочной лестничной параметризации со встроенными 2D CORDIC-модулями, которая дает возможность реализации умножителя кватернионов как обратимого оператора умножения кватернионов в целочисленной арифметике. Данную схему можно рассматривать как структурное решение для цифровой обработки медиаданных без потерь, когда данные и алгоритмы представляются в алгебре кватернионов.

УДК 004.31-027.31(076.5)
ББК 32.973.26-04я73

ISBN 978-985-543-448-2

Содержание

Введение

Последние 20 лет в цифровой обработке сигналов алгебра кватернионов рассматривается как новая парадигма, позволяющая осуществлять обработку сигналов непосредственно в многомерном домене. Кватернионы уже с успехом использовались в цифровой обработке сигналов, например в адаптивных фильтрах Калмана, оценке спектра по методу MUSIC, сингулярной декомпозиции матриц, параунитарных банках фильтров (ПУБФ) – Q-ПУБФ и других приложениях. ПУБФ могут рассматриваться как наиболее значимое преобразование среди многоскоростных систем цифровой обработки сигналов для систем мультимедиа. Это обусловлено тем фактом, что подобные банки фильтров являются преобразованиями без потерь в дополнение к гарантированной перфективной реконструкции сигнала. Точное соотношение между энергиями во всем частотном диапазоне и в субполосах сильно упрощает теоретические выкладки и делает ПУБФ полезными в применении к компрессии (кодированию) изображений.

Базовым элементарным преобразованием Q-ПУБФ является умножение кватернионов, где один из сомножителей – кватернион-константа, т. е. кватернион с постоянными параметрами. При этом умножение кватернионов – ключевая операция, от эффективной реализации которой зависят характеристики всего преобразования. Однако прямое умножение матрицы на вектор потребует 16 умножений действительных чисел и 12 алгебраических сложений. Продукт умножения кватернионов можно вычислить на основе восьми действительных умножений, но данная техника не подходит для случая, когда один операнд есть кватернион-константа. Известен алгоритм выполнения умножения кватернионов на основе лестничной структуры в арифметике с фиксированной запятой, вычислительная сложность которого составляет 12 операций умножения на действительные числа. Здесь основной компонент схемы – умножитель-накопитель. Реализация умножителя в этом случае требует больших затрат компонент структуры ПЛИС типа FPGA. Представляют интерес вычислительные схемы умножителя кватернионов без использования устройств действительных умножений.

Умножитель кватернионов на распределенной арифметике позволяет получать сбалансированные схемные решения как по производительности, так и по потребляемой мощности. Однако для достижения высокой скорости и точности вычислений здесь необходим большой объем памяти.

В настоящее время в области систем мультимедиа сделан большой прогресс в развитии алгоритмов и архитектурных решений для высокопроизводительных вычислений с малыми аппаратными затратами – CORDIC-алгоритмов (COrdinate Rotation DIgital Computer). Красота CORDIC заключается в том, что с помощью простых операций сдвига и сложения $a \pm b2^{-i}$ можно вычислять ряд задач во многих областях, таких как обработка сигналов и изображений, системах телекоммуникаций, робототехнике, 3D компьютерной графике. Параллельные и конвейерные структуры CORDIC-процессоров позволяют достигать высокой скорости вычислений.

В пособии сформулирована задача оптимизации схемных решений CORDIC-алгоритма для фиксированного угла поворота как для 2D-пространства, так и многомерного (4D) евклидового вращения. Результатом решения задачи является оптимизированное множество микровращений для фиксированного и известного угла поворота. При этом итерации микровращения и масштабирования выполняются последовательно друг за другом на одном аппаратном модуле. Эффективные CORDIC-схемы получаются благодаря тому, что параметры сдвига представляются нелинейной функцией числа сдвигов входных операндов операции микровращения, а не являются последовательностью однобитных сдвигов как в стандартном CORDIC-алгоритме. В предложенной структуре параметры схемы подбираются каждый раз под соответствующий кватернион-константу, на который выполняется умножение, т. е. для каждого постоянного сомножителя – кватерниона константы получается разное количество сдвигов и разные масштабные множители, а также свой набор операторов направления

вращения, которые должны храниться в памяти. Это цена за оптимизацию итерационного процесса и общая проблема с подходами построения более гибких CORDIC-алгоритмов.

1 БАЗОВАЯ ТЕХНИКА CORDIC

1.1 2D CORDIC-алгоритм

Вычислительный алгоритм CORDIC, предложенный Волдером и позднее развитый Вальтером, представляет собой алгоритм, оперирующий двумерными (2D) векторами для вычисления вращений, используя простые арифметические примитивы: сдвиг и сложение. Поворот вектора $\mathbf{x} = [x_1 \; x_2]^T$ на угол φ может быть описан произведением данного вектора на соответствующую матрицу вращения $\mathbf{R}_2(\varphi)$:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{bmatrix}}_{\mathbf{R}_2(\varphi)} \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_{\mathbf{x}}. \tag{1.1}$$

С другой стороны это выражение описывает произведение комплексного числа $(x_1 + jx_2)$ на комплексную экспоненту $e^{-j\varphi}$. CORDIC-алгоритм аппроксимирует матрицу вращения произведением N элементарных вращений $\mathbf{R}_2(i)$:

$$\mathbf{R}_2(\varphi) = \prod_{i=0}^{N-1} \mathbf{R}_2(i), \tag{1.2}$$

$$\mathbf{R}_2(i) = \left(1 + \delta t_i^2\right)^{-1/2} \mathbf{U}_2(i), \; \mathbf{U}_2(i) = \begin{bmatrix} 1 & -\delta \cdot \sigma(i) t_i \\ \delta \cdot \sigma(i) t_i & 1 \end{bmatrix}, \tag{1.3}$$

где $\mathbf{U}_2(i)$ – не нормализованная часть $\mathbf{R}_2(i)$, $\delta = 1$, если вращение в евклидовом пространстве, $\delta = -1$, если вращение осуществляется в псевдоевклидовом пространстве (гиперболическое вращение), $t_i = 2^{-i}$. Параметры N, $\sigma(i)$, i, подбираются таким образом, что результат произведения (1.2) аппроксимирует вращение на угол

$$\varphi \approx \sum_{i=0}^{N-1} \sigma(i) \cdot 2^{-i}, \tag{1.4}$$

где N – число вращений, а параметр $\sigma(i) \in \{-1,1\}$ определяет направление вращения для оставшейся части угла.

CORDIC-алгоритм состоит из двух процессов: итерационного процесса вращения и процесса масштабирования. На каждой итерации входной вектор $\mathbf{x}$ поворачивается на угол 2^{-i}, но результат вращения не соответствует фиксированному радиусу – длина вектора увеличивается на величину $\left(1+2^{-2i}\right)^{1/2}$, т. е. необходима нормализация размера вектора, которая выполняется в процессе масштабирования. Таким образом, на i-й итерации для соответствующего входного вектора $\mathbf{x}(i) = \begin{bmatrix} x_1(i) & x_2(i) \end{bmatrix}^T$ выходной вектор $\mathbf{x}(i+1) = \begin{bmatrix} x_1(i+1) & x_2(i+1) \end{bmatrix}^T$ вычисляется следующим образом:

$$\mathbf{x}(i+1) = \mathbf{U}_2(i) \cdot \mathbf{x}(i), \tag{1.5}$$

$$\varphi(i+1) = \varphi(i) - \sigma(i)2^{-i}, \quad \text{для } i = \overline{0, N-1}, \tag{1.6}$$

где $\mathbf{U}_2(i)$ представляет собой оператор микровращения на i-й итерации CORDIC-алгоритма; $\varphi(i)$ – оставшийся угол после $(i-1)$-й итерации.

Структура модуля CORDIC алгоритма (1.5) и (1.6) показана на рисунке 1.1.

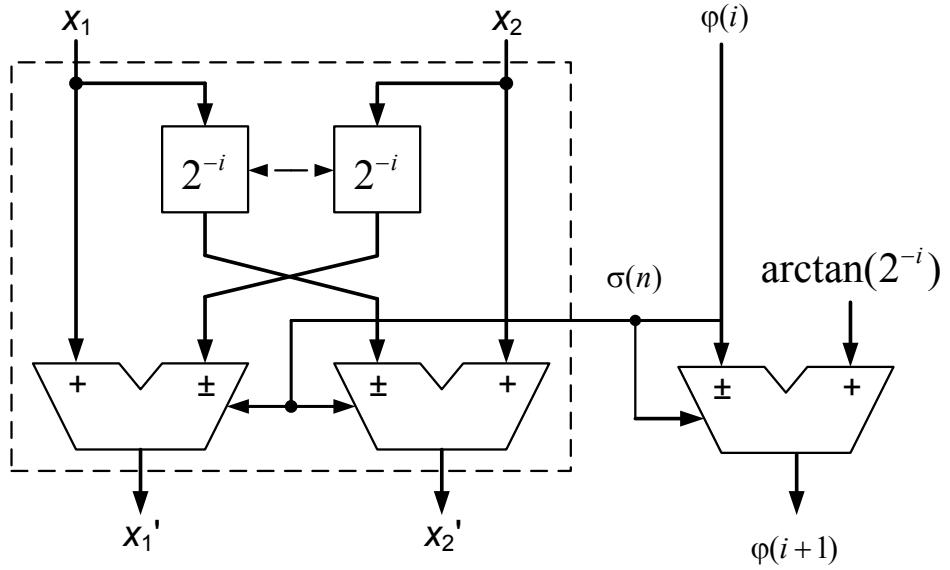


Рисунок 1.1 – Структура модуля CORDIC-алгоритма

Амплитуда результата нормализуется на этапе масштабирования после окончания итерационного этапа: выходной вектор $\mathbf{x}(N)$ делится на масштабный фактор $T = \prod_{i=0}^{N-1}\left(1+2^{-2i}\right)^{1/2}$. Итак, окончательный результат $\mathbf{x}_{out}(N)$ CORDIC-алгоритма может быть представлен как

$$\mathbf{x}_{out}(N) = \frac{1}{T}\mathbf{x}(N) = \left(\prod_{i=0}^{N-1}\left(1+2^{-2i}\right)^{1/2}\right)^{-1}\mathbf{x}(N). \qquad (1.7)$$

Таким образом, масштабирование требует дополнительной вычислительной операции, но, если масштабный фактор представить в виде

$$\frac{1}{T} = \sum_{s=0}^{S-1}\sigma(s)2^{-i(s)}, \qquad (1.8)$$

где $\sigma(s) = \pm 1$ и $0 \le i(s) \le N-1$, тогда процесс масштабирования может быть описан рекурсивным выражением: пусть $\mathbf{x}(0) = 0$ и $\mathbf{x}(S) = \mathbf{x}_{out}(N)$, тогда

$$\mathbf{x}(s+1) = \mathbf{x}(s) + \sigma(s)2^{-i(s)}\mathbf{x}(N) \quad \text{для } s = 0,\ 1,\ \ldots,\ S-1. \qquad (1.9)$$

В стандартной CORDIC-технике параметры алгоритма равны: $\sigma(i) \in \{-1,1\}$, $i = \overline{0, N-1}$ и $N = B$, где B – число разрядов в двоичном представлении компонент векторов, т. е. сдвиг и знак оператора направления вращения для обоих процессов как итерационного, так и масштабирования определяются в режиме онлайн на основе входных данных. Выбор соответствующих параметров алгоритма определяет фиксированное число микровращений (итераций). В приложении А показан расчет базовых углов и управляющих параметров 2D CORDIC-алгоритма.

1.2 Многомерный CORDIC-алгоритм

Известны быстрые 3D и 4D CORDIC-алгоритмы, число итераций которых немного больше, чем для одного 2D CORDIC-вращения. Данные алгоритмы основываются на фундаментальной теореме Кэли. Формула Кэли устанавливает

связь между ортогональными и кососимметрическими операторами в евклидовом пространстве и позволяет любое nD евклидово вращение $\mathbf{R}_n$, для которого характеристическое число равно –1, представить как

$$\mathbf{R}_n = \left(\mathbf{I}_n - \mathbf{T}_n\right)\left(\mathbf{I}_n - \mathbf{T}_n\right)^{-1}, \tag{1.10}$$

где $\mathbf{I}_n$ – единичная матрица $n \times n$; $\mathbf{T}_n$ – кососимметрическая матрица (оператор), т. е. $\mathbf{T}_n^T = -\mathbf{T}_n$.

Например, для i-го 3D элементарного вращения матрица $\mathbf{T}_3(i)$ конструируется следующим образом:

$$\mathbf{T}_{3,i} = \begin{bmatrix} 0 & -\sigma_1(i)t & \sigma_2(i)t \\ \sigma_1(i)t & 0 & -\sigma_1(i)t \\ -\sigma_2(i)t & \sigma_1(i)t & 0 \end{bmatrix}, \sigma_1(i), \sigma_2(i) \in \{1, -1\},$$

где $t = 2^{-i}$. Соответствующая элементарная нормализованная матрица вращения согласно (1.10) определится как

$$\mathbf{R}_3(i) = (\mathbf{I}_3 - \mathbf{T}_{3,i})(\mathbf{I}_3 + \mathbf{T}_{3,i})^{-1} =$$
$$= \frac{1}{\sqrt{1+3\cdot t^2}} \begin{bmatrix} 1-t^2 & 2(\sigma_1(i)t + t^2) & -2(\sigma_2(i)t + t^2) \\ -2(\sigma_1(i)t + t^2) & 1-t^2 & 2(\sigma_1(i)t + t^2) \\ 2(\sigma_2(i)t + t^2) & -2(\sigma_1(i)t + t^2) & 1-t^2 \end{bmatrix}.$$

Данные вращения выполняются относительно оси $\begin{bmatrix}\sigma_1(i) & \sigma_2(i) & \sigma_1(i)\end{bmatrix}^T$, что соответствует направляющему вектору либо $\begin{bmatrix}1 & 1 & 1\end{bmatrix}^T$, либо $\begin{bmatrix}1 & -1 & 1\end{bmatrix}^T$. Когда вектор $\begin{bmatrix}x_1 & x_2 & x_3\end{bmatrix}^T$ следует привести к направлению $\begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$, т. е. первой координатной оси, то знаки направления вращения на i-й итерации $\sigma_1(i)$ и $\sigma_2(i)$ выбираются согласно следующему закону управления: $\sigma_1(i) = \text{sign}\left(x_{1,i-1} \cdot x_{2,i-1}\right)$, $\sigma_2(i) = -\text{sign}\left(x_{1,i-1} \cdot x_{2,i-1}\right)$, где $x_{j,i-1}$ определяет j-ю компоненту вектора в начале

i-й итерации. Главный недостаток данного метода состоит в том, что он не является систематическим подходом конструирования nD CORDIC элементарных матриц вращения $\mathbf{R}_n$ для произвольных n.

CORDIC-алгоритм может быть расширен на многомерный случай с использованием преобразования Хаусхолдера, матрица отражения которого определяется следующим образом:

$$\mathbf{H}_n = \mathbf{I}_n - 2\frac{\mathbf{u}\cdot\mathbf{u}^T}{\mathbf{u}^T\cdot\mathbf{u}},$$

где $\mathbf{u}$ – n-мерный вектор.

Результат преобразования ($\mathbf{H}_n \cdot \mathbf{v}$) есть отражение n-мерного вектора $\mathbf{v}$ по отношению к гиперплоскости с нормалью $\mathbf{u}$, которая проходит через начало координат. Таким образом, CORDIC-алгоритм на основе отражения Хаусхолдера выполняет операцию векторизации n-мерного вектора к одной из осей. Ненормализованная матрица вращения для n-мерного вектора соответствующей i-й итерации $\mathbf{R}_{Hn}(i)$ определяется как произведение двух простых отражений Хаусхолдера:

$$\mathbf{R}_{Hn}(i) = \left(\mathbf{I}_n - 2\frac{\mathbf{e}_1\mathbf{e}_1^T}{\mathbf{e}_1^T\mathbf{e}_1}\right)\left(\mathbf{I}_n - 2\frac{\mathbf{u}_i\mathbf{u}_i^T}{\mathbf{u}_i^T\mathbf{u}_i}\right), \qquad (1.11)$$

где $\mathbf{e}_1 = \begin{bmatrix}1 & 0 & \dots & 0\end{bmatrix}^T$ и $\mathbf{u}_i = \begin{bmatrix}1 & \sigma_1(i)\cdot t_i & \dots & \sigma_{n-1}(i)\cdot t_i\end{bmatrix}^T$, $t_i = \tan\varphi_i = 2^{-i}$, а управляющие сигналы направления вращения выбираются как:

$$\sigma_j(i) = \operatorname{sign}\left(x_{1,i-1}\cdot x_{j+1,i-1}\right),\ 1 \le j \le n-1.$$

Для $n = 4$ нормализованная матрица вращения $\mathbf{R}_{H4}(i)$ согласно (1.11) задается как функция сдвигов и управляющих сигналов направления вращения:

$$\mathbf{R}_{H4}(i) = \frac{1}{\sqrt{1+3\cdot t_i^2}} \begin{bmatrix} 1 & -\sigma_1(i)t_i & -\sigma_2(i)t_i & -\sigma_3(i)t_i \\ \sigma_1(i)t_i & 1 & \sigma_3(i)t_i & -\sigma_2(i)t_i \\ \sigma_2(i)t_i & -\sigma_3(i)t_i & 1 & \sigma_1(i)t_i \\ \sigma_3(i)t_i & \sigma_2(i)t_i & -\sigma_1(i)t_i & 1 \end{bmatrix}. \quad (1.12)$$

Следовательно, Хаусхолдер CORDIC-алгоритм представлен как обобщение CORDIC-алгоритма на *n*D-мерный случай для осуществления вращения в евклидовом и псевдо-евклидовом пространствах. Следует так же отметить, что *n*D CORDIC-алгоритм хорошо отображается на параллельных структурах процессоров.

2 Оптимизация параметров встроенного 2D CONST CORDIC-модуля

Имеется много алгоритмов, например, дискретное косинусное преобразование или быстрое преобразование Фурье, где параметры преобразования известны априори. Задача выбора параметров схемы умножителя кватернионов со встроенным 2D CORDIC-алгоритмом наиболее сложная часть проекта. Как отмечалось выше, для фиксированного угла вращения φ с целью достижения заданной точности за минимальное число итераций M параметры управления для итерационного и масштабирующего процессов CORDIC-алгоритма могут быть определены и оптимизированы заранее во время проектирования схемы CORDIC-алгоритма:

- фиксированный угол φ предопределяет некоторое множество базисных углов CORDIC-алгоритма $\{\varphi(i) = \arctan(2^{-\tau(i)})$ для $0 \le i \le M-1\}$. Тогда, чтобы повернуть вектор $\left[x_1 x_2\right]^T$ на угол φ для получения вектора $\left[x_1^{out} x_2^{out}\right]^T$, CORDIC-алгоритм (1.2)–(1.12) может быть модифицирован следующим образом:

$$\begin{bmatrix} x_1(i+1) \\ x_2(i+1) \end{bmatrix} = \begin{bmatrix} 1 & -\sigma(i)2^{-\tau(i)} \\ \sigma(i)2^{-\tau(i)} & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1(i) \\ x_2(i) \end{bmatrix}, \quad \begin{bmatrix} x_1^{out} \\ x_2^{out} \end{bmatrix} \leftarrow \frac{1}{K} \begin{bmatrix} x_1(M) \\ x_2(M) \end{bmatrix}; \qquad (2.1)$$

- масштабный фактор $1/K$ зависит от множества базисных углов $\{\varphi(i)\}$, а точность алгоритма определяется как близко фактический угол вращения φ_A аппроксимирует заданный φ за M итераций микровращений. Это в свою очередь определяет отклонение фактического вектора вращения от требуемого.

Произвольное значение масштабирующего фактора $1/K = \prod_{i=0}^{M-1}\left(1 + 2^{-2\tau(i)}\right)^{-1/2}$ при $(1/K < 1)$ может быть аппроксимировано выражением

$$\frac{1}{K} \approx \prod_{l=0}^{L-1}\left[1 - 2^{-s(l)}\right], \qquad (2.2)$$

где L и $s(l) \in \mathbb{N} \setminus \{0\}$ выбраны как продукт аппроксимации $1/K$, а процесс масштабирования описывается следующим рекурсивным выражением:

$$\mathbf{x}(l+1) = \mathbf{x}(l) - 2^{-s(l)}\mathbf{x}(l), \quad l = 0,\ 1,\ \ldots,\ L-1,\ \mathbf{x}(0) = \mathbf{x}(M),\ \mathbf{x}(L) = \mathbf{x}_{out}(M). \qquad (2.3)$$

Таким образом, для фиксированного угла вращения параметры управления для итерационного и масштабирующего процессов CORDIC-алгоритма могут быть оптимизированы за два шага:

1. Приведение угла к заданным ограничениям. Преобразования (1.2), (1.3) или (2.1), (2.2) применимы только для вращений на углы, для которых $\cos\varphi \geq 0$ и $|\cos\varphi| \geq |\sin\varphi|$, т. е. для углов $|\varphi| \leq \frac{\pi}{4}$. Так для осуществления вращения на произвольный угол φ, не удовлетворяющий данным условиям, всегда может быть реализовано простой пре- и пост- обработкой матрицы вращений $\mathbf{R}_2$ с подходящим углом $\tilde{\varphi}$:

$$\mathbf{R}_2(\varphi) = \mathbf{P}_{post}^{\text{CORDIC}}\mathbf{R}_2(\tilde{\varphi})\mathbf{P}_{pre}^{\text{CORDIC}}, \qquad (2.4)$$

в частности,

$$\mathbf{R}_2(\varphi) = \mathbf{\Gamma}_2\mathbf{R}_2(-\varphi)\mathbf{\Gamma}_2,\ \mathbf{R}_2(\varphi) = \mathbf{J}_2\mathbf{R}_2(-\varphi)\mathbf{\Gamma}_2,\ \mathbf{R}_2(\varphi) = -\mathbf{J}_2\mathbf{R}_2(\pi-\varphi)\mathbf{J}_2, \qquad (2.5)$$

т. е. достаточно поменять местами величины и/или их знаки. Здесь $\mathbf{J}_2$ – обратная единичная матрица 2×2, а $\mathbf{\Gamma}_2 = \text{diag}(1,-1)$.

2. Минимизация числа базисных углов во множестве $\{\varphi(i)\}$ для достижения требуемой точности вычисления преобразования: определяются параметры $\tau(i)$, M и $s(l)$, L.

Простой псевдокод программы алгоритма расчета операторов сдвига $\tau(i)$ и направления вращения $\sigma(i)$ для схемы 2D CONST Q-CORDIC для фиксированного угла вращения показан ниже. Для заданной максимальной точности ε_φ, которая определяется как максимальная ошибка $\mathbf{\Delta}_\varphi$ между матрицей вращения $\mathbf{M}$,

определенной заданным углом – входным параметром $\mathbf{q} = \left[q_1, q_2\right]^T$, и ее аппроксимацией $\mathbf{IR}_2(i)$. Алгоритм оптимизации поиска параметров $\tau(i)$ и $\sigma(i)$ базируется на минимизации целевой функции $\max\left|\mathbf{\Delta}_\varphi\right|$. Последовательность матриц приближения $\mathbf{IR}_2(i),\ i = 0,\ 1,\ 2...$ определяется операцией CORDIC, для которой матрица вращения формируется из параметров управления $\tau(i)$ и $\sigma(i)$, рассчитанных на i-й итерации. Для $i = 0$ существует единичная матрица вращения вектора, лежащего на оси абсцисс, вокруг центра декартовых координат на угол, определенный вектором $\mathbf{q} = \left[q_1, q_2\right]^T$ $(\mathrm{Re}\, q = q_1, \mathrm{Im}\, q = q_2)$. При этом входной вектор $\mathbf{x}(i) = \left[x_1(i), x_2(i)\right]^T$ равен комплексно-сопряженному вектору $\overline{\mathbf{q}} = \left[q_1, -q_2\right]^T$ $(|\mathbf{q}| = |\overline{\mathbf{q}}|)$.

Алгоритм 1. Расчета параметров управления $\tau(i)$ и $\sigma(i)$ схемы
2D CONST CORDIC

Вход: $\mathbf{q} = \left[q_1, q_2\right]^T, (\mathrm{Re}\, q = q_1, \mathrm{Im}\, q = q_2)$ – комплексное число-константа, т.е. фиксированный угол вращения;

ε_φ – точность аппроксимации заданного угла вращения

Выход: $\tau(i)$ – неотрицательное целое число сдвигов на i-й итерации;

$\sigma(i) \in \pm 1$ – направление вращения на i-й итерации

Начало:

1. Сформировать матрицы вращения $\mathbf{M} = \begin{bmatrix} q_1 & -q_2 \\ q_2 & q_1 \end{bmatrix}$ и единичную матрицу $\mathbf{IR}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

2. Установить: $i = 0$, $\mathbf{x}(i) = \overline{\mathbf{q}}$, $(\mathbf{x}(i) = \left[x_1(i), x_2(i)\right]^T)$ и $\mathbf{IR}_2(i) = \mathbf{IR}_2$.

3. Определить значение параметра направления вращения: $\sigma(i) = -\mathrm{sign}[x_2(i)]$.

4. Вычислить параметр числа сдвигов: $\tau(i) = \text{round}(\log_2 |x_2(i)/x_1(i)|)$.

5. Сохранить $\sigma(i)$ и $\tau(i)$ как параметры i-й итерации.

6. Сформировать $\chi = 1 + j\sigma(i)2^{\tau(i)}$, $j = \sqrt{-1}$.

7. Выполнить итерацию аппроксимации:

$\mathbf{IR}_2(i+1) = \mathbf{R}_2(\chi)\mathbf{IR}_2(i)$, где $\mathbf{R}_2(\chi)$ – матрица вращения (1.3)

8. Нормализовать результат итерации: $\mathbf{IR}_2(i+1) = \mathbf{IR}_2(i+1)/|\chi|$.

9. Вычислить ошибку аппроксимации $\mathbf{\Delta}_\varphi = \mathbf{IR}_2(i+1) - \mathbf{M}$.

10. Если $\max|\mathbf{\Delta}_\varphi| \le \varepsilon_\varphi$, то $M = i$, иначе выполнить итерацию операции CORDIC $\mathbf{x}(i+1) = \mathbf{R}_2(\chi)\mathbf{x}(i)$ и $i = i+1$, перейти к пункту **3**.

Конец.

Алгоритм 2 описывает задачу оптимизации поиска параметров L и $s(l)$ процесса масштабирования схемы 2D CONST CORDIC. Масштабный множитель, обусловленный выполнением итерационного процесса с параметрами $\tau(i)$ и $\sigma(i)$, которые получены в результате выполнения Алгоритма 1, вычисляется как $1/K = \prod_{i=0}^{M-1}\left(1 + 2^{-2\tau(i)}\right)^{-1/2}$. Значение целевой функции ΔK определяется отличием от единицы отношения масштабного множителя $1/K$ к его аппроксимированному значению $(1/K)_A$, т. е. $\Delta K = |(1/K)/(1/K)_A - 1|$. Алгоритм начинает работу с вычисления аппроксимированного значения масштабного множителя для одного сдвига, а затем число масштабирующих сдвигов увеличивается на единицу пока ΔK больше, чем заданная точность ε_K операции масштабирования, величина которой одного порядка как ε_φ в Алгоритме 1, потому что ошибки ΔK и $\mathbf{\Delta}_\varphi$ вносят равный вклад в общую ошибку схемы 2D CONST CORDIC.

Алгоритм 2. Расчета параметров L и $s(l)$ процесса масштабирования

Вход: $1/K = \prod_{i=0}^{M-1}\left(1+2^{-2\tau(i)}\right)^{-1/2}$ – масштабный множитель;

ε_K – точность операции масштабирования.

Выход: L и $s(l) \in \mathbb{N} \setminus \{0\}$ - неотрицательное целое число сдвигов на l-й итерации процесса масштабирования.

Начало:

1. Установить $A = 1$, $k = 1$, $l = 0$

2. Вычислить $(1/K)_A = A(1-2^{-k})$

3. Если $(1/K)_A \geq (1/K)$, то сохранить число сдвигов на l-й итерации процесса масштабирования $s(l) = k$ и $A = (1/K)_A$, иначе $k = k+1$ и переход к пункту **2**

4. Оценить точность аппроксимации $\Delta K = \left|(1/K)/(1/K)_A - 1\right|$.

5. Если $\Delta K \leq \varepsilon_K$, то $L = k$, иначе переход к п.**2**

Конец

В отличие от стандартного CORDIC-алгоритма это позволяет операцию масштабирования осуществлять независимо от угла φ, когда определение угла есть часть CORDIC-алгоритма, а параметры $\tau(i)$, M и L представлять произвольными значениями. Например, выбор количества сдвигов $\tau(i) \in \mathbb{N}$ не зависит от номера итерации i, но при этом может быть получена требуемая точность за меньшее число микровращений.

На рисунке 2.1 показаны схемы реализации (а) итерационного процесса и (б) процесса масштабирования 2D CONST CORDIC-алгоритма для априори известного угла (коэффициента-константы комплексного умножения). Анализ данных схемных решений показывает, что итерационный и масштабирующий процессы могут быть реализованы на одной коммутируемой во времени

CORDIC-схеме. При этом затраты оборудования будут минимальны, а производительность данного процессора будет обратно пропорциональна сумме числа микровращений и итераций процесса масштабирования.

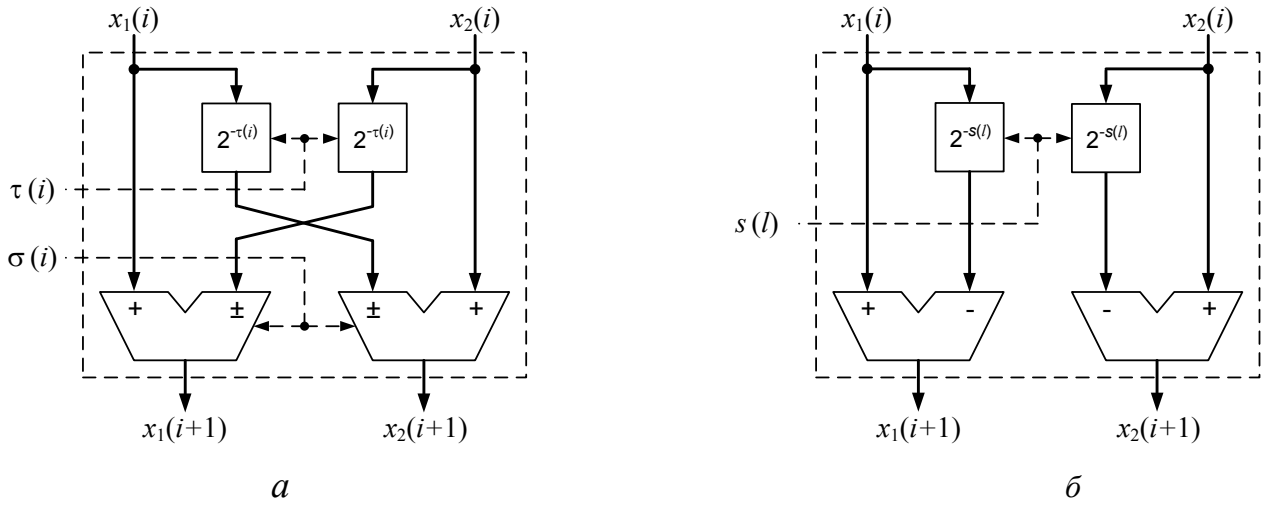


Рисунок 2.1 – Схемы для реализации (а) микровращений и (б) итераций масштабирования 2D CONST CORDIC алгоритма для априори известного угла.

С целью увеличения скорости работы 2D CORDIC-процессора вычисления (2.1) и (2.3) могут осуществляться на конвейерной CORDIC-схеме. Структурная схема операции микровращения 2D CONST CORDIC показана на рисунке 2.2.

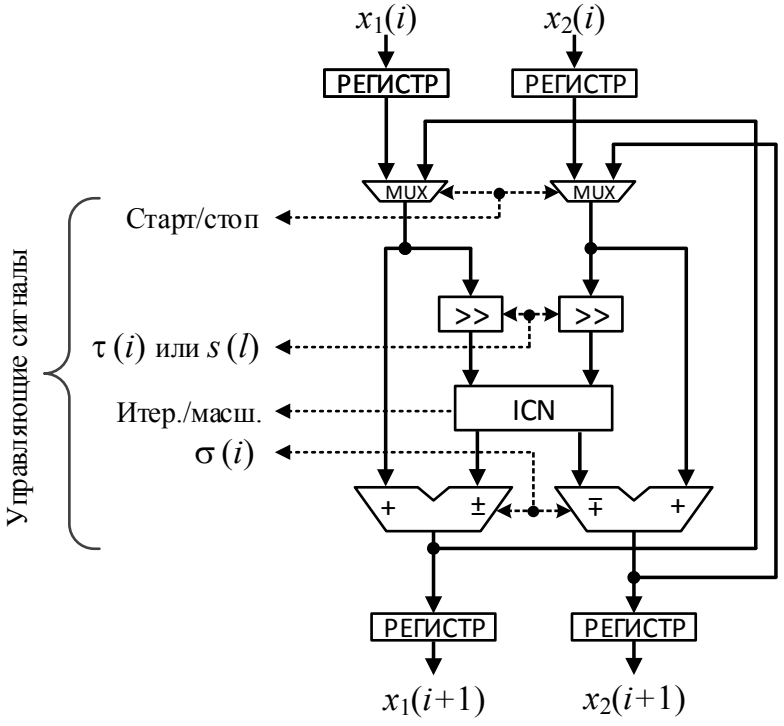


Рисунок 2.2 – Структурная схема универсальной операции 2-D CONST CORDIC

Ядро схемы образуют два сумматора, работающие в режиме суммирования и вычитания, два регистра сдвига вправо и модуль коммутации ICN, основное назначение которого – переключать выходы регистров сдвига в соответствии с режимами работы схемы 2D CONST CORDIC: итерационный процесс (выходы коммутируются накрест), масштабирующий процесс (выходы коммутируются прямо). Компоненты $x_1(i)$ и $x_2(i)$ загружаются во входные регистры и передаются в сумматоры согласно схеме рисунке 2.2. Содержимое сумматоров $x_1(i+1)$, $x_2(i+1)$ загружается во входные регистры для следующей CORDIC-итерации. Время выполнения операции микровращения складывается из задержки на входных мультиплексорах t_{MUX}, операции сдвига t_{Sh}, а также задержки в модуле коммутации ICN t_{MUX} и работы сумматора/вычитателя t_{ADD}. Полное время выполнения операции вращения на заданный угол складывается из времен итерационного и масштабирующего процессов:

$$t_{\mu R}^{S} = (M + L)(2t_{MUX} + t_{Sh} + t_{ADD}), \tag{2.6}$$

где M и L – число итераций итерационного и масштабирующего процессов соответственно.

Для повышения производительности модуля 2D CORDIC можно использовать конвейерное включение схем микровращения 2D CONST CORDIC. При этом латентность операции будет определяться числом схем микровращения в конвейере, но пропускная способность возрастет во столько же раз. На рисунке 2.3 показана структура многоступенчатого конвейера операции микровращения 2D CONST CORDIC. Время критического пути в данном случае равно $t_{Sh} + t_{ADD} + t_{BF}$, где t_{BF} – время записи в буферные регистры (БР). Длина конвейера (число ступеней микровращения и масштабирования) определяется количеством итераций итерационного и масштабирующего процессов. Специализация ступеней конвейера на выполнение соответствующей операции (итерационного или масштабирующего процессов) позволяет исключить модуль коммутации ICN.

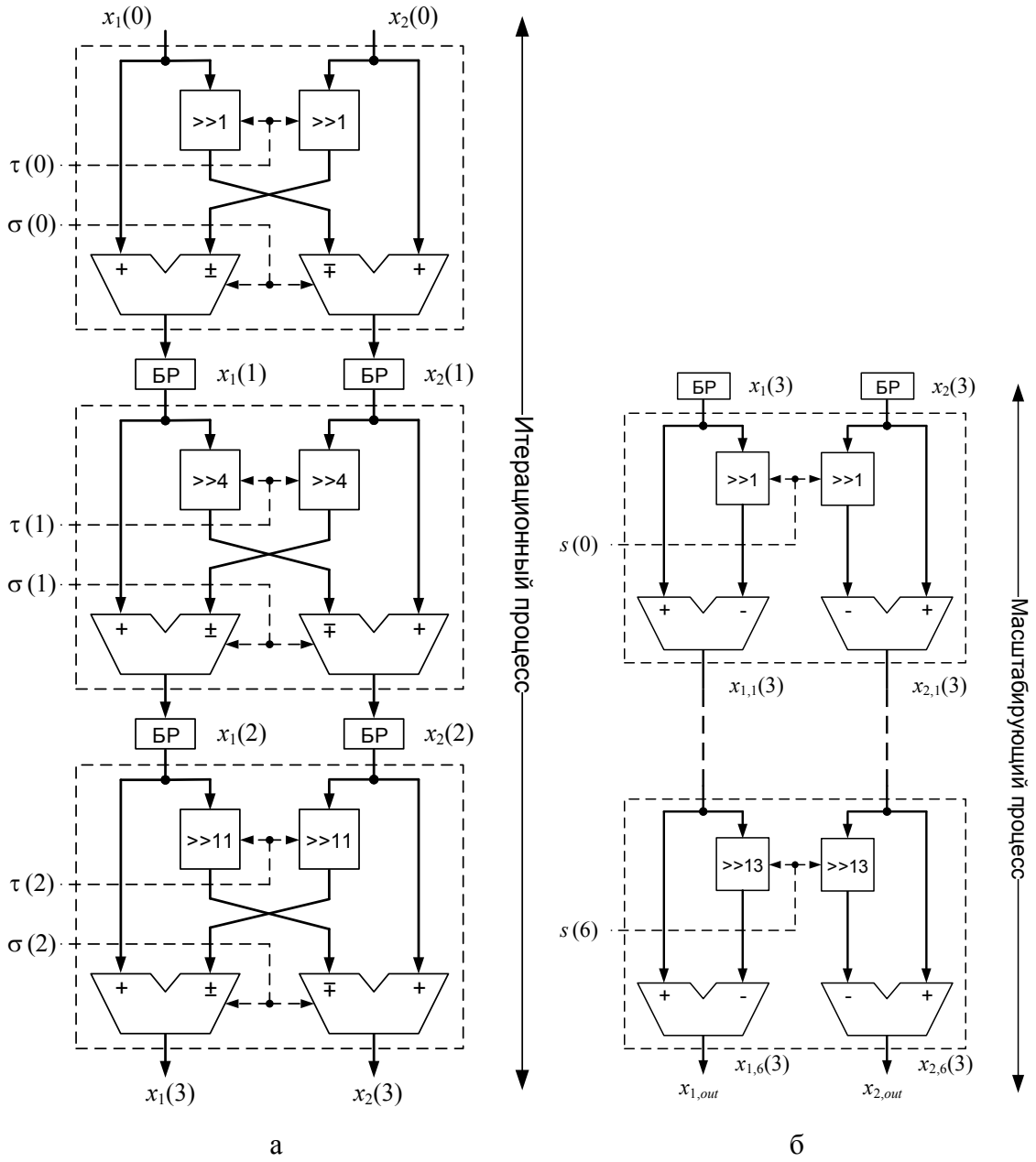
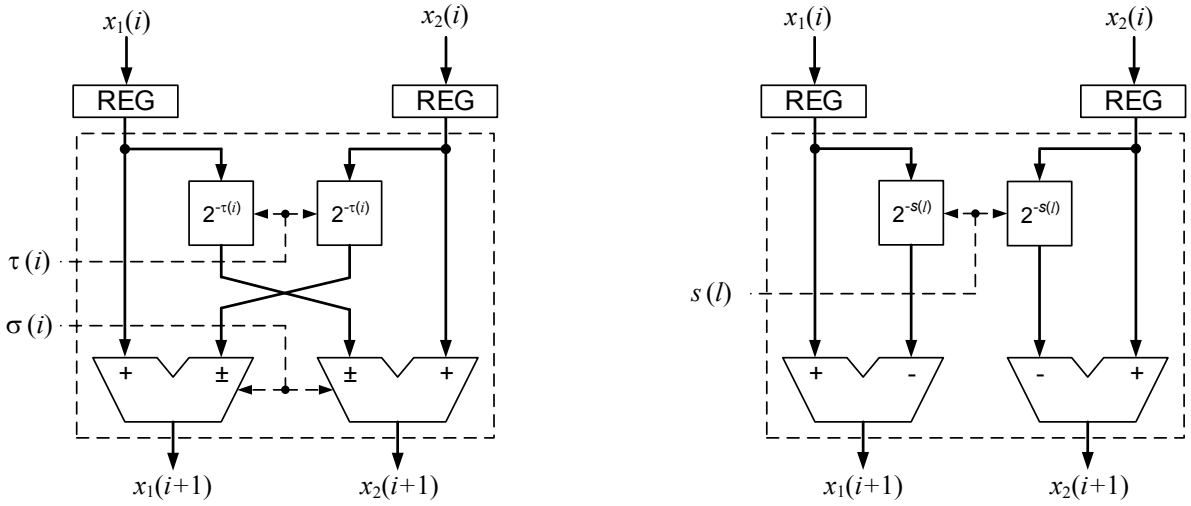


Рисунок 2.3 – Структурная схема многоступенчатого конвейера 2D CONST CORDIC

Для того, чтобы для любой итерации итерационного процесса или масштабирующего процесса время операции микровращения было постоянным необходимо применять регистры сдвига с параллельной организацией сдвига на произвольное число битов. Данное обстоятельство обуславливает повышенное использование аппаратного ресурса, например, ПЛИС с архитектурой FPGA. Так как параметры сдвига рассчитаны заранее, то можно сдвиг осуществить напря-

мую путем соответствующей коммутации выходов буферных регистров со входами сумматоров следующей ступени операции микровращения/процесса масштабирования конвейера (рисунок 2.4).



а б

Рисунок 2.4 – Схема ступени многоступенчатого конвейера 2D CONST CORDIC

Таким образом, каскадное включение таких специализированных ступеней упрощает структуру поточного процессора операции 2D CONST CORDIC, а также обеспечивает высокую пропускную способность реализации конвейерного процессора умножителя на кватернион-константу (рисунок 2.5). Время критического пути в данном случае соответствует только сумме одной операции сложения/вычитания t_{ADD} и задержке на буферном регистре t_{BF}, т. е.

$$t_{\mu R}^{P} = t_{ADD} + t_{BF}. \tag{2.7}$$

Следовательно, пропускная способность поточного процессора операции 2D CONST CORDIC приблизительно в $(M + L)$ раз выше, чем у последовательной рекурсивной схемы операции микровращения 2D CONST CORDIC согласно рисуноку 2.2. Латентность выполнения операции микровращения 2D CONST CORDIC в мультиступенчатом конвейере поточного процессора составит величину

$$t_{LAT} = (M + L)(t_{ADD} + t_{BF}). \tag{2.8}$$

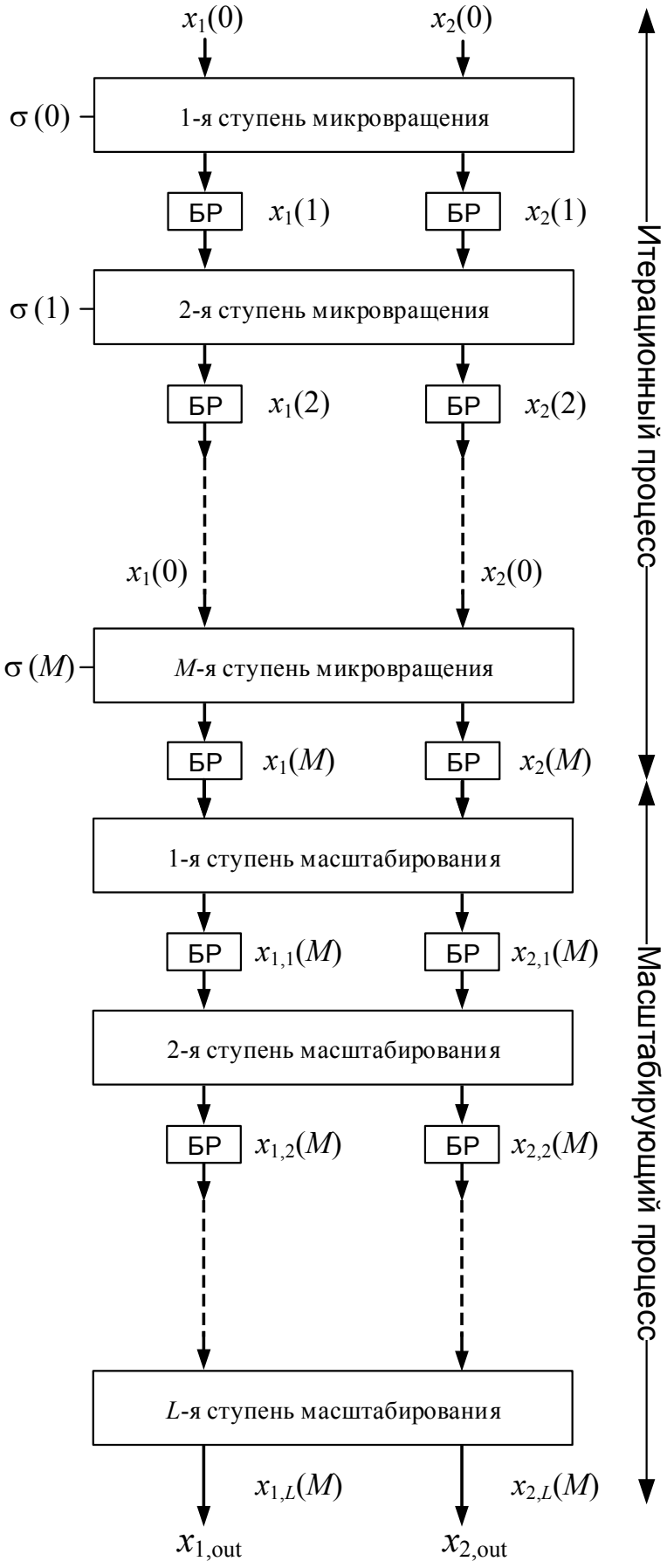


Рисунок 2.5 – Конвейерный процессор 2D CONST CORDIC

3 4D CORDIC-алгоритм умножения на кватернион константу

3.1 Постановка задачи

Алгебра кватернионов $\mathbb{H}$ является ассоциативной некоммутативной четырёхмерной алгеброй $\mathbb{H}=\{\mathbf{q}=q_1+q_2i+q_3j+q_4k \mid q_1,q_2,q_3,q_4\in\mathbb{R}\}$, где ортогональные мнимые части подчиняются следующим законам умножения: $i^2=j^2=k^2=ijk=-1$, $ij=-ji=k$, $jk=-kj=i$, $ki=-ik=j$. Операции сложения и умножения кватернионов могут быть реализованы с использованием векторно-матричной алгебры. Для этого кватернионы представляются в виде четырех мерных векторов. Умножение кватернионов в векторной форме производится по правилу умножения вектора на матрицу, т. к. операция умножения некоммутативна, то продукт умножения определяется операторами умножения «слева» $\mathbf{M}^{+}(q)$ и «справа» $\mathbf{M}^{-}(q)$:

$$qx \Leftrightarrow \underbrace{\begin{bmatrix} q_1 & -q_2 & -q_3 & -q_4 \\ q_2 & q_1 & -q_4 & q_3 \\ q_3 & q_4 & q_1 & -q_2 \\ q_4 & -q_3 & q_2 & q_1 \end{bmatrix}}_{\mathbf{M}^{+}(q)} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}}_{\mathbf{x}}, \quad xq \Leftrightarrow \underbrace{\begin{bmatrix} q_1 & -q_2 & -q_3 & -q_4 \\ q_2 & q_1 & q_4 & -q_3 \\ q_3 & -q_4 & q_1 & q_2 \\ q_4 & q_3 & -q_2 & q_1 \end{bmatrix}}_{\mathbf{M}^{-}(q)} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}}_{\mathbf{x}}. \tag{3.1}$$

Обе матрицы (3.1) ортогональные до масштабного коэффициента, заданного нормой кватернионов $|q|=\sqrt{q_1^2+q_2^2+q_3^2+q_4^2}$: $\mathbf{M}^{\pm}(q)^{-1}=(1/|q|)\mathbf{M}^{\pm}(q)^T$, где ^{T} – знак транспонирования матрицы. Как следует из структур матриц, их свойства не зависят от значений компонент кватерниона и, следовательно, не чувствительны к их модификации, например, квантованию. Матрицы $\mathbf{M}^{+}(q)$ и $\mathbf{M}^{-}(q)$ соотносятся между собой как

$$\mathbf{M}^{\mp}(q)=\mathbf{D}_{\mathrm{C}}\mathbf{M}^{\pm}(q)^T\mathbf{D}_{\mathrm{C}}, \tag{3.2}$$

где $\mathbf{D}_{\mathrm{C}}=\mathrm{diag}(1,-\mathbf{I}_3)$ описывает в матричной нотации оператор гиперкомплексного сопряжения. Определив сопряженный кватернион $\overline{q}=q_1-q_2i-q_3j-q_4k$ в векторно-матричном виде как $\overline{\mathbf{q}}=\mathbf{D}_{\mathrm{C}}\mathbf{q}$, можно получить, что

$$\mathbf{M}^{\pm}(\overline{q}) = \mathbf{M}^{\pm}(q)^{T}, \tag{3.3}$$

тогда выражение (3.2) переопределяется следующим образом

$$\mathbf{M}^{\mp}(\overline{q}) = \mathbf{D}_{\mathrm{C}}\mathbf{M}^{\pm}(q)\mathbf{D}_{\mathrm{C}}. \tag{3.4}$$

Последнее равенство показывает, что результаты, полученные для умножения «левого» qx или «правого» типа xq, могут быть применимы к умножению на сопряженный кватернион. В частности, это доказывает, что вычислительная сложность операторов умножений кватернионов одинакова. Это обстоятельство позволяет сосредоточить внимание на организации вычисления продукта qx оператора «левого» умножения $\mathbf{M}^{+}(q)$.

В операции умножения кватернионов (3.1) операнд x является входной переменной, а коэффициент q – кватернион-константа, компоненты которого определяют характеристики, например, Q-ПУБФ, и задают матрицу умножения. Оба операнда нормированы (норма равна единице), а их матрицы умножения различаются от заданных только масштабным коэффициентом. Для данных кватернионов операция умножения на сопряженный кватернион $\overline{q}$ эквивалентно умножению на обратный кватернион:

$$\begin{aligned} & q \cdot \overline{q} = q \cdot q^{-1} = 1 + (0i + 0j + 0k) \Leftrightarrow \\ & \Leftrightarrow \ \mathbf{M}^{\pm}(q)\underbrace{\begin{bmatrix} q_1 & -q_2 & -q_3 & -q_4 \end{bmatrix}^{T}}_{\overline{\mathbf{q}}^{T}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^{T}}_{\mathbf{e}_1^{T}}. \end{aligned} \tag{3.5}$$

Это матричное выражение может интерпретироваться с различных точек зрения: результат умножения на $\overline{q}$ является кватернионом, у которого мнимые части обнулены, что эквивалентно расположению вектора $\overline{\mathbf{q}}$ вдоль первой канонической оси при умножении на $\mathbf{M}^{\pm}(q)$. Таким образом, единичные кватернионы, у которых матрицы умножения – ортонормальные версии матриц умножения оригинальных кватернионов, могут быть использованы для вывода CORDIC-алгоритма умножения кватернионов на основе факторизаций $\mathbf{M}^{\pm}(q)$,

которые имеют аналогичный эффект умножению на вектор $\overline{\mathbf{q}}$ при определённых ограничениях:

Во-первых, все сомножители в факторизации должны иметь структуру аналогичную матрице $\mathbf{M}^{\pm}(q)$, т. е. произведение любого их множества представляет матрицу умножения кватернионов соответствующего типа:

$$\begin{aligned}\mathbf{M}^{+}(q_{k-1}\cdot\ldots\cdot q_0)&=\mathbf{M}^{+}(q_{k-1})\cdot\ldots\cdot\mathbf{M}^{+}(q_0),\\ \mathbf{M}^{-}(q_0\cdot\ldots\cdot q_{k-1})&=\mathbf{M}^{-}(q_{k-1})\cdot\ldots\cdot\mathbf{M}^{-}(q_0)\end{aligned} \tag{3.6}$$

Во-вторых, сомножители в факторизации (3.6) должна сходиться к единичной матрице 4×4, $\mathbf{I}_4$, т. е. возможная бесконечная серия сомножителей в факторизации может быть усечена для некоторой аппроксимации исходной матрицы при допустимой ошибке;

В-третьих, сомножители в факторизации должны быть «простыми» матрицами, умножение на которые можно реализовать с использованием нескольких операций сдвига и сложений.

3.2 4D CORDIC алгоритм умножения кватернионов для сингулярного разложения матриц

В 4D CORDIC-алгоритме умножения кватернионов для сингулярного разложения матриц на основе преобразования Хаусхолдера матрицы умножения (3.1) можно представить в виде следующей факторизации:

$$\mathbf{M}^{\pm}(q)\approx\prod_{n=0}^{N-1}\frac{1}{\left|\zeta\left(\tau(i),\sigma_1(i),\sigma_2(i),\sigma_3(i)\right)\right|}\mathbf{M}^{\pm}\left(\zeta\left(\tau(i),\sigma_1(i),\sigma_2(i),\sigma_3(i)\right)\right). \tag{3.7}$$

где кватернион $\mathbf{q}$ на i-й итерации 4D CORDIC алгоритма аппроксимируется как $\zeta\left(\tau(i),\sigma_1(i),\sigma_2(i),\sigma_3(i)\right)=1+\sigma_1(i)2^{\tau(i)}i+\sigma_2(i)2^{\tau(i)}j+\sigma_3(i)2^{\tau(i)}k$, а его матрица умножения, которая описывает i-ю итерацию 4D CORDIC-алгоритма, представляется следующим образом:

$$\mathbf{M}^{\pm}\left(\zeta(\tau(i),\sigma_1(i),\sigma_2(i),\sigma_3(i))\right)\approx\begin{bmatrix}1 & -\sigma_1(i)2^{\tau(i)} & -\sigma_2(i)2^{\tau(i)} & -\sigma_3(i)2^{\tau(i)}\\ \sigma_1(i)2^{\tau(i)} & 1 & \mp\sigma_3(i)2^{\tau(i)} & \pm\sigma_2(i)2^{\tau(i)}\\ \sigma_2(i)2^{\tau(i)} & \pm\sigma_3(i)2^{\tau(i)} & 1 & \mp\sigma_1(i)2^{\tau(i)}\\ \sigma_3(i)2^{\tau(i)} & \mp\sigma_2(i)2^{\tau(i)} & \pm\sigma_1(i)2^{\tau(i)} & 1\end{bmatrix}. \quad (3.8)$$

Параметр количества двоичных сдвигов $\tau(i)$ зависит от индекса итераций i, т. е. $\tau(i)=-i$, а параметры управления направлением вращения $\sigma_1(i),\sigma_2(i),\sigma_3(i)$ принимают значения 1 или –1.

Таким образом, 4D CORDIC-алгоритм умножения кватернионов с матрицей умножения (3.8) определяется как

$$\mathbf{x}_{i+1}=\mathbf{M}^{\pm}\left(\zeta\left(\tau(i),\sigma_1(i),\sigma_2(i),\sigma_3(i)\right)\right)\cdot\mathbf{x}_i,(i=\overline{0,N-1};\mathbf{x}_0=\mathbf{x}). \quad (3.9)$$

Для того, чтобы 4D вектор $\mathbf{x}$ привести к направлению первой канонической оси сигналы управления направлением вращения на i-й итерации определяются кватернионом $\mathbf{x}_i$:

$$\begin{aligned}\sigma_1(i)&=-\text{sign}\left(x_{1,i}\right)\cdot\text{sign}\left(x_{2,i}\right),\\ \sigma_2(i)&=-\text{sign}\left(x_{1,i}\right)\cdot\text{sign}\left(x_{3,i}\right),\\ \sigma_3(i)&=-\text{sign}\left(x_{1,i}\right)\cdot\text{sign}\left(x_{4,i}\right).\end{aligned} \quad (3.10)$$

Норма кватерниона $\mathbf{x}_{i+1}$ или длина вектора на i-й итерации согласно (1.12) равна $\left|\zeta(\tau(i),\sigma_1(i),\sigma_2(i),\sigma_3(i))\right|=\sqrt{1+3\cdot2^{-2\tau(i)}}$. Следовательно, выходной результат итерационного процесса $\mathbf{x}_N$ нормализуется на коэффициент $1/K=\prod_{i=0}^{N-1}1/\sqrt{1+3\cdot2^{-2\tau(i)}}$, который может быть аппроксимирован в соответствии с (1.10), и тогда умножение на данный коэффициент реализуется раздельно от итерационного процесса как серия сдвигов и сложений (процесс масштабирования). Окончательный результат 4D CORDIC-алгоритма равен $\mathbf{x}_{out,N}=(1/K)\mathbf{x}_N$. Анализ матрицы (3.8) показывает, что при увеличении числа итераций i матрица $\mathbf{M}^{\pm}\left(\zeta\left(\tau(i),\sigma_1(i),\sigma_2(i),\sigma_3(i)\right)\right)$ сходится к единичной матрице 4×4.

Схема вычисления итерации (микровращений) 4D CORDIC алгоритма может быть реализована на четырёх регистрах сдвига и на четырех 4-х входовых сумматорах (см. рис. 3.1). Это обуславливает необходимость дополнительной логики контроля переполнения, а при реализации на ПЛИС типа FPGA такие сумматоры требуют повышенный аппаратный ресурс.

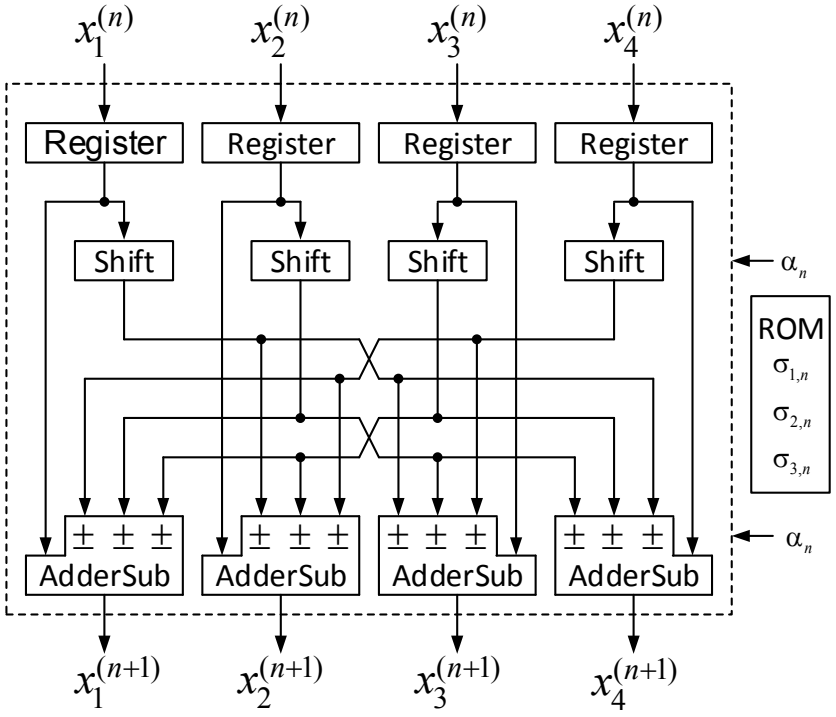


Рисунок 3.1 – Схема вычисления итерации (микровращений) 4D CORDIC-алгоритма

Схема микровращений 4D CORDIC-алгоритма не может быть использована на этапе масштабирования результата итерационного процесса, 4D-итерации (3.9), (3.10) не всегда сходятся. Как показали экспериментальные исследования процент кватернионов на множестве 10^4 случайно сгенерированных единичных кватернионов, для которых заданная точность не достижима (алгоритм не сходится), изменялся от 6 до 18 %, для точности CORDIC-процесса соответственно 10^{-2} и 10^{-6}.

3.3 4D CORDIC-алгоритм умножения кватернионов (4D CONST Q-CORDIC) на базе 2D-вращений в 4D-гиперплоскости

Таким образом, необходим поиск альтернативных Q-CORDIC-алгоритмов, которые просто проектируются, требуют меньше вычислительных ресурсов и непременно сходятся. Предлагается конструировать 4D CORDIC-алгоритм умножения кватернионов на базе 2D-вращений в 4D-гиперплоскости, основываясь на факторизации (3.6) матрицы умножения кватернионов $\mathbf{M}^{\pm}(q)$, когда единичный кватернион q представляется как произведение $\left(q = d_k d_j d_i\right)$ трех вырожденных единичных кватернионов d_i, d_j, d_k, у которых соответственно одна отличная от нуля мнимая часть i, j, k:

$$\begin{aligned} \mathbf{M}^{+}(q) &= \mathbf{M}^{+}(d_k)\cdot\mathbf{M}^{+}(d_j)\cdot\mathbf{M}^{+}(d_i), \\ \mathbf{M}^{-}(q) &= \mathbf{M}^{-}(d_i)\cdot\mathbf{M}^{-}(d_j)\cdot\mathbf{M}^{-}(d_k) \end{aligned} \tag{3.11}$$

Матрицы умножения вырожденных кватернионов $\mathbf{M}^{\pm}(d_i)$, $\mathbf{M}^{\pm}(d_j)$ и $\mathbf{M}^{\pm}(d_k)$ разреженные, каждая из которых описывает 2D-вращение в различных плоскостях 4D гиперплоскости на одинаковый угол, но возможно в противоположных направлениях:

$$\begin{aligned} \mathbf{M}^{\pm}(d_i) &= \mathrm{diag}(\mathbf{R}_2(\varphi_i), \mathbf{R}_2(\pm\varphi_i)), \\ \mathbf{M}^{\pm}(d_j) &= \mathbf{P}_{1324}\,\mathrm{diag}(\mathbf{R}_2(\varphi_j), \mathbf{R}_2(\mp\varphi_j))\mathbf{P}^{T}{}_{1324}, \\ \mathbf{M}^{\pm}(d_k) &= \mathbf{P}_{1423}\,\mathrm{diag}(\mathbf{R}_2(\varphi_k), \mathbf{R}_2(\pm\varphi_k))\mathbf{P}^{T}{}_{1423}, \end{aligned} \tag{3.12}$$

где $\mathbf{R}_2(\varphi)$ – матрица вращения Гивенса (1.1), а $\mathbf{P}_{1324}$ и $\mathbf{P}_{1423}$ – матрицы перестановок строк и столбцов в матрице умножения кватернионов в соответствии с индексами.

Так как умножение на каждую разряженную матрицу умножения вырожденного кватерниона равносильно выполнению вращения на одинаковые углы, которые могут осуществляться параллельно на двух 2D CORDIC-блоках для фиксированного угла вращения φ, поэтому итерации плотно связанных

2-D CORDIC-алгоритмов идентифицируются с одной итерацией 4D CORDIC-алгоритма. Изменение плоскостей 2D-вращений или переназначения входов и выходов 2D CORDIC-блоков эквивалентно переключению 4D CORDIC-алгоритмов среди трех итерационных процессов, описываемых следующими разреженными матрицами умножения вырожденных кватернионов:

$$\mathbf{M}^{\pm}(\zeta(\tau(i),\sigma_1(i),0,0)) \approx \begin{bmatrix} 1 & -\sigma_1(i)2^{-\tau(i)} & 0 & 0 \\ \sigma_1(i)2^{-\tau(i)} & 1 & 0 & 0 \\ 0 & 0 & 1 & \pm\sigma_1(i)2^{-\tau(i)} \\ 0 & 0 & \pm\sigma_1(i)2^{-\tau(i)} & 1 \end{bmatrix}, \quad (3.13)$$

$$\mathbf{M}^{\pm}(\zeta(\tau(i),0,\sigma_2(i),0)) \approx \begin{bmatrix} 1 & 0 & -\sigma_2(i)2^{-\tau(i)} & 0 \\ 0 & 1 & 0 & \pm\sigma_2(i)2^{-\tau(i)} \\ \sigma_2(i)2^{-\tau(i)} & 0 & 1 & 0 \\ 0 & \pm\sigma_2(i)2^{-\tau(i)} & 0 & 1 \end{bmatrix}, \quad (3.14)$$

$$\mathbf{M}^{\pm}(\zeta(\tau(i),0,0,\sigma_3(i))) \approx \begin{bmatrix} 1 & 0 & 0 & -\sigma_3(i)2^{-\tau(i)} \\ 0 & 1 & \pm\sigma_3(i)2^{-\tau(i)} & 0 \\ 0 & \pm\sigma_3(i)2^{-\tau(i)} & 1 & 0 \\ \sigma_3(i)2^{-\tau(i)} & 0 & 0 & 1 \end{bmatrix}. \quad (3.15)$$

Переходя от размерности пространства 4D к 2D и обратно умножение кватернионов может быть аппроксимировано дискретными вырожденными кватернионами, у которых действительные части равны единице, а не нулевые мнимые части представляются отрицательной степенью двух.

В факторизации (3.11), (3.12) выбор углов $\varphi_i, \varphi_j, \varphi_k$ определяет соответствующую тройку вырожденных единичных кватернионов d_i, d_j, d_k, умножение которых на сопряженный кватернион $\overline{q}$ обнуляет мнимые части результирующего произведения, т. е. $d_i \cdot d_j \cdot d_k \cdot \overline{q} = q \cdot \overline{q} = 1 + 0i + 0j + 0k$. Обнуление мнимых частей произведения на основе CORDIC-алгоритма предполагает решение задачи аппроксимации данного процесса, используя дискретные вырожденные кватернионы (3.13)–(3.15), каждый из которых выбирается следующим образом:

$$d_i = \cos(\varphi_i) + i\sin(\varphi_i), \quad \varphi_i = \operatorname{arctg}(q_2/q_1) \to d_i\overline{q} = 0i + \cdots,$$

$$d_j = \cos(\varphi_j) + j\sin(\varphi_j), \varphi_j = \operatorname{arctg}(q_3/q_1) \to d_j\overline{q} = \cdots + 0j + \cdots,$$

$$d_k = \cos(\varphi_k) + j\sin(\varphi_k), \varphi_k = \operatorname{arctg}(q_4/q_1) \to d_k\overline{q} = \cdots + 0k.$$

После умножения на сопряженный кватернион мнимые части могут быть отличны от нуля, тогда данный процесс повторяется для другой тройки дискретных вырожденных кватернионов и так до тех пор, пока мнимые части не будут обнулены или, по крайней мере, их значения не будут меньше некоторой заданной величины. Это объясняется тем, что преобразование ортогональное и не модифицирует норму входных кватернионов, то если одна из мнимых частей увеличивается, тогда другая мнимая часть должна уменьшаться, т. к. суммы квадратов мнимых частей единичного кватерниона равны единице минус квадрат действительной части. Таким образом, последовательность умножений на дискретные вырожденные кватернионы путем увеличения реальной части неявно уменьшают значения мнимых частей, даже если их максимальная амплитуда временно увеличивается. Обнуление мнимых частей может осуществляться последовательно, начиная с мнимой части по i, однако, фактически необходимо на каждой итерации выбирать для обнуления мнимую часть, у которой наибольшая величина. Хотя оба подхода сходятся, но у второго, как показали экспериментальные исследования, сходимость процесса выше. Максимум точности операции умножения на кватернион-константу достигается как близко за некоторое число итераций на заданном наборе управляющих параметров $\tau(i)$ и $\sigma(i)$ значения мнимых компонент по модулю вектора $\mathbf{x}(i)$ операции CORDIC на сопряженный кватернион-константу $\overline{q}$ близки к нулю. Псевдокод программы алгоритма расчета операторов сдвига $\tau(i)$ и направления вращения $\sigma(i)$ для 4D CONST Q-CORDIC-алгоритма умножения кватернионов показан ниже.

Алгоритм 3 Расчёта параметров управления схемы 4D CONST Q-CORDIC умножения кватернионов (кватернион q – константа, т. е. угол вращения – фиксированный)

Вход: $\overline{\mathbf{q}}$ – сопряженный кватернион-константа;

ε – точность операции CORDIC

Выход: $\tau(i)$ – неотрицательное целое число, т. е. число сдвигов на i-й итерации;

$\sigma(i) \in \pm 1$ – направление вращения на i-й итерации.

Начало:

1. Установить**:** $i = 0$ и $\mathbf{x}(i) = \overline{\mathbf{q}}$.

2. Определить индекс $m(i)$ наибольшей мнимой части i-й итерации:

$x_{m(i)}(i) = \max\limits_{2 \le l \le 4} \left| x_l(i) \right|$.

3. Определить значение параметра направления вращения:

$\sigma(i) = -\text{sign}(x_{m(i)}(i))$.

4. Вычислить параметр числа сдвигов: $\tau(i) = \text{round}(\log_2 \left| x_{m(i)}(i) \right|)$.

5. Сохранить $m(i), \sigma(i)$ и $\tau(i)$ как параметры i-й итерации.

6. Установить $\zeta = 1 + 0i + 0j + 0k$ и $\zeta_{m(i)} = \sigma(i) 2^{\tau(i)}$.

7. Выполнить итерацию операции CORDIC, чтобы вычислить:

$\mathbf{x}(i+1) = \mathbf{M}^{+}(\zeta)\mathbf{x}(i)$**.**

8. Нормализовать результат итерации: $\mathbf{x}(i+1) = \mathbf{x}(i+1) / \left| \zeta \right|$.

9. Если $\forall_{l=2...4} \left| x_l(i+1) \right| \le \varepsilon$, то переход на «**Конец»**, иначе $i = i + 1$ и перейти к пункту **2**.

Конец

В таблице 3.1 приведен пример расчета параметров управления схемы 4D CONST Q-CORDIC умножения на кватернион константу $q = \left[\ 4, -1,\ 3, -2\right] / \sqrt{30}$ для точности CORDIC-операции $\varepsilon < 10^{-3}$.

Таблица 3.1 – Пример расчёта коэффициентов для кватерниона $q = \left[4, -1, 3, -2\right] / \sqrt{30}$

i	$m(i)$	$\tau(i)$	$\sigma(i)$	β_2	β_1	Значение вектора $\mathbf{x}(i)$
0	–	–	–	–	–	0.730297, 0.182574, –0.547723, 0.365148
1	2	1	+1	1	0	0.903696, 0.387298, 0.129099, 0.129099
2	1	2	–1	0	1	0.981495, –0.057735, 0.173205, 0.057735
3	2	3	–1	1	0	0.994199, –0.070014, –0.070014, 0.042008
4	2	4	+1	1	0	0.995206, 0.053842, –0.074684, 0.033000
5	1	4	+1	0	1	0.996784, 0.057519, 0.049333, 0.026067
6	1	5	–1	0	1	0.998431, –0.004770, 0.050863, 0.022939
7	3	5	–1	1	1	0.999659, –0.006192, –0.011517, 0.022596
8	2	5	–1	1	0	0.999877, –0.006549, –0.011318, –0.008639
9	2	7	+1	1	0	0.999932, –0.006683, 0.004305, –0.008535
10	3	7	+1	1	1	0.999943, –0.006749, 0.004200, 0.007088
11	3	8	–1	1	1	0.999968, –0.006716, 0.004253, –0.000724
12	2	8	–1	1	0	0.999990, 0.001096, 0.004258, –0.000691
13	2	9	+1	1	0	0.999993, 0.001101, –0.003554, –0.000682
14	1	10	–1	0	1	0.999999, 0.001099, 0.000352, –0.000687
15	1	11	+1	0	1	0.999999, –0.000855, 0.000351, –0.000687

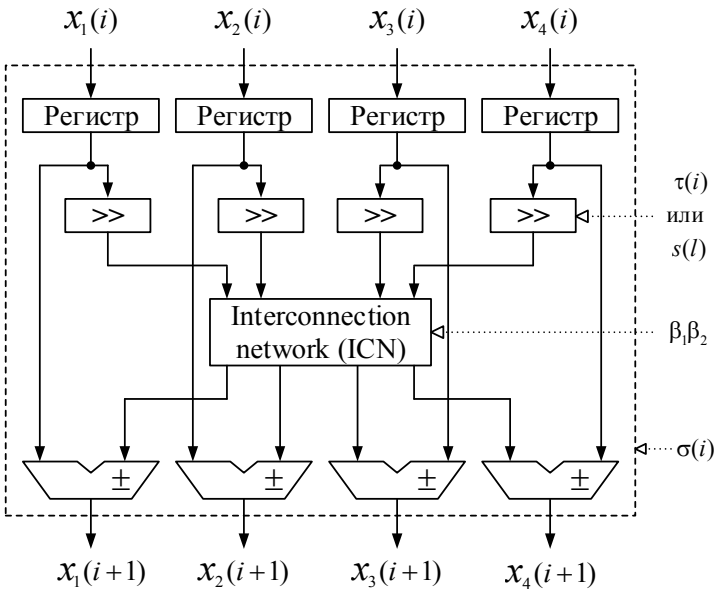
В предложенном многомерном CORDIC-алгоритме умножения кватернионов микровращения соответствуют умножению вырожденных кватернионов, у которых только одна мнимая часть отличная от нуля и может быть разной для разных CORDIC-итераций. С другой стороны, параметры знака и сдвига итерации выбираются таким образом, чтобы итерации могли пропускаться или повторяться. Как видно из таблицы 3.1 сходимость алгоритма достигается. Исследования проводились на множестве 10^6 случайно сгенерированных единичных кватернионов, точность операции CORDIC задавалась в диапазоне от 10^{-2} до 10^{-6}. Среднее число итераций колебалось от 9 до 28, а максимальное число итераций

от 15 до 41. Масштабный множитель здесь меньше, чем у 4D Q-CORDIC-алгоритма и для случая равного количества итераций для каждой разреженной матрицы определяется как $1/K = \left[\prod_{i=0}^{N-1} 1/\sqrt{1+3\cdot 2^{-2\tau(i)}}\right]^3$.

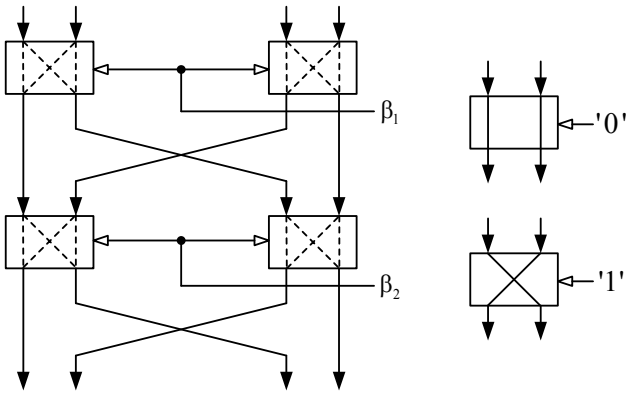
3.4 Схема микровращений алгоритма 4D CONST Q-CORDIC

Схема микровращений алгоритма 4-D CONST-Q-CORDIC умножения кватернионов может быть реализована как показано на рисунке 3.2, а, используя четыре двухвходовых сумматора и схему коммутации ICN (interconnection network), с помощью которой выбирается не нулевая мнимая часть для текущей итерации. Центральное место схемы 4-D CONST-Q-CORDIC алгоритма занимает схема коммутации ICN (см. рисунок 3.2, б). Исходя из структуры разряженных матриц умножения вырожденных кватернионов (3.13)–(3.15) достаточно четыре варианта перестановок данных, поэтому можно воспользоваться упрощенной сетью Бенесса. Выбирая режимы работы переключающих схем 2×2 между «прямо» и «накрест» (см. рисунок 3.2, в), можно получить четыре возможных потока перестановки данных (рисунок 3.3).

Экспериментальные исследования 4D CONST Q-CORDIC-алгоритм проводились для аппаратной реализации схемы микровращений на FPGA XC6VLX240T (Xilinx Virtex-6). Для 16 разрядных данных задействованы следующее количество ресурсов FPGA: LUT6-FF – 189, LUT – 105, FF – 81, максимальная частота задающего генератора составляет 315 МГц.

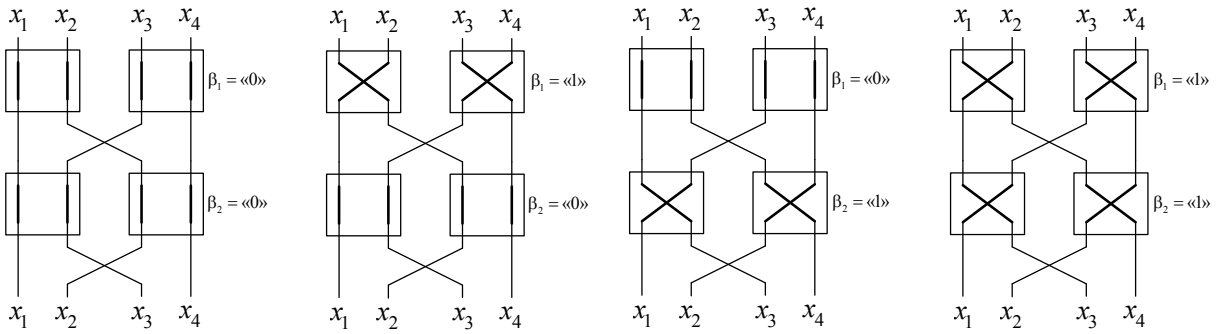


а



б в

Рисунок 3.2 – 4D CONST Q-CORDIC умножения кватернионов: а – схема микровращений алгоритма 4D CONST Q-CORDIC; б – схема коммутации ICN для перестановки данных на основе сети Бенеса; в – состояния переключающих схем



а б в г

Рисунок 3.3 – Возможные варианты перестановки компонент входного вектора $\mathbf{x}(i)$ на схеме коммутации ICN для перестановки данных: а – путь передачи данных без изменения их порядка; б – операции с мнимой частью i; в – операции с мнимой частью j; операции с мнимой частью k

Кроме того, если в ICN выбирать путь передачи данных без изменения их порядка (см. рисунок 3.3, а), тогда схема микровращений 4D CONST Q-CORDIC-алгоритма умножения кватернионов может быть использована для реализации этапа масштабирования, который представляется как умножение на вырожденный кватернион с не нулевой действительной компонентой:

$$\mathbf{M}^{\pm}(1-2^{-s(l)}+0i+0j+0k)=\begin{bmatrix}1-2^{-s(l)} & 0 & 0 & 0\\ 0 & 1-2^{-s(l)} & 0 & 0\\ 0 & 0 & 1-2^{-s(l)} & 0\\ 0 & 0 & 0 & 1-2^{-s(l)}\end{bmatrix}.$$

Здесь $1-2^{-s(l)}$ – аппроксимация масштабного множителя l-й итерации процесса масштабирования, расчет которого показан в разделе 2. На рисунке 3.4 приведены временные диаграммы работы схемы микровращений 4D CONST Q-CORDIC алгоритма умножения на кватернион $q=[4,-1,\ 3,-2]/\sqrt{30}$.

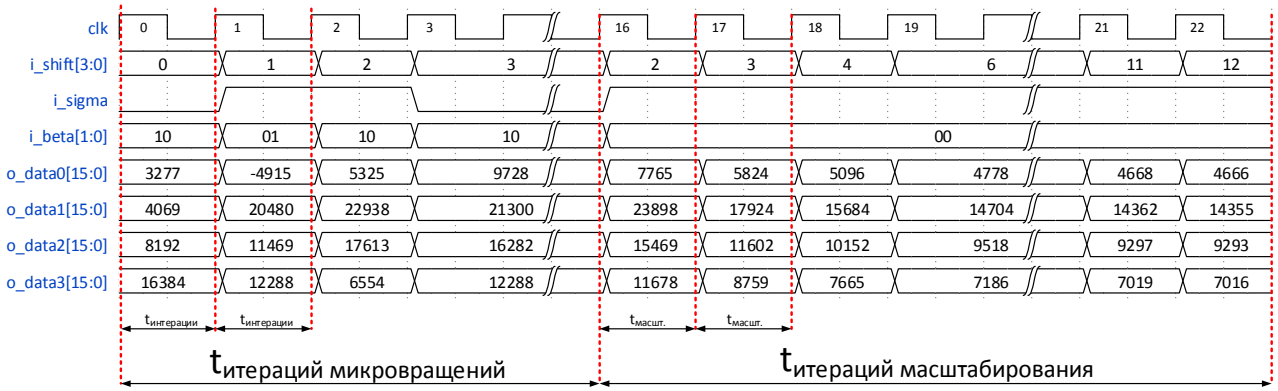


Рисунок 3.4 – Временные диаграммы работы схемы микровращений 4D CONST Q-CORDIC-алгоритма умножения на кватернион $q=[4,-1,3,-2]/\sqrt{30}$

Весь цикл работы состоит из последовательно выполняемых шагов итерационного (16 шагов) и масштабирующего (7 шагов) процессов, выбор которых определяется настройкой схемы ICN(β_1,β_2) (значение кода i_beta (1:0)). Направление вращения определяет управляющий сигнал i_sigma($\sigma(i)$), а количество

сдвигов – i_shift ($\tau(i)$). Переключение схемы коммутации ICN для выбора соответствующей разреженной матрицы умножения вырожденного кватерниона задает значение кода i_beta (β_1, β_2). Результат операции микровращения формируется на выходах i_data0 ($x_1(i+1)$), i_data1 ($x_2(i+1)$), i_data2 ($x_3(i+1)$), i_data3 ($x_4(i+1)$). Как следует из временной диаграммы, скорость выполнения операции умножения кватернионов обратно пропорциональна суммарному времени итерационного и масштабирующего процессов.

4 Целочисленный умножитель на кватернион константу на основе встроенного 2D CORDIC-модуля

4.1 Блочная лестничная схемная параметризация оператора умножения кватернионов

Ограничение динамического диапазона арифметики с фиксированной запятой приводит к тому, что результаты умножений округляются и, следовательно, реализовать обратное преобразование (умножение на сопряженный кватернион) в случае единично нормированных коэффициентов в арифметике с фиксированной запятой не представляется возможным, получается только аппроксимация входной величины. Это является недопустимым для таких приложений, например, как компрессия изображений по схеме L2L – lossless-to-lossy (сжатие и восстановление изображений как без потерь, так и с контролируемым внесением артефактов). Данный недостаток может быть исправлен, если перейти к лестничной схемной параметризации оператора умножения кватернионов, которая позволяет выполнять обратимые «целое к целому» отображения, используя арифметику с фиксированной запятой. Вычислительная сложность операции составит 12 умножений на действительные числа. Однако, здесь не берется во внимание число операций округления результатов умножений в лестничной структуре, рассматриваемые как белый шум, число которых влияет на компактность энергии в субполосах Q-ПУБФ. Следовательно, количество операций округления должно быть уменьшено в максимально возможной степени для приложений кодирования без потерь. На рисунке 4.1 показана структура обратимого преобразования на основе блочной лестничной схемной параметризации, которая представляет собой специальный класс структур обратимых преобразователей с лестничной схемной параметризацией. Количество источников шума здесь сокращается за счет объединения многих операций округления.

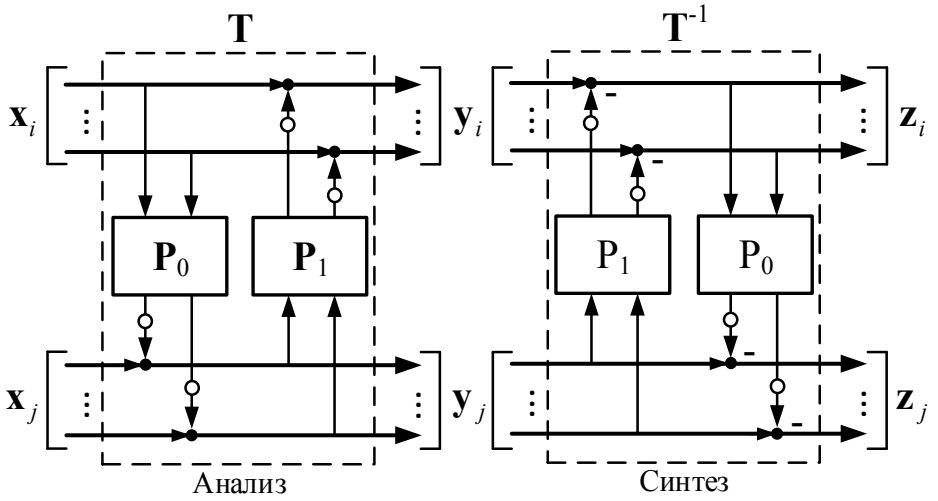


Рисунок 4.1 – Структура обратимого преобразователя на основе блочной лестничной схемной параметризации (черные и белые кружки обозначают операции суммирования и округления соответственно)

Векторы $\mathbf{x}_i$ и $\mathbf{x}_j$ на входе и на выходе $\mathbf{y}_i$ и $\mathbf{y}_j$ прямого преобразования, а также векторы $\mathbf{z}_i$ и $\mathbf{z}_j$ на выходе обратного преобразования и блоки преобразования – матрицы $\mathbf{P}_0$ и $\mathbf{P}_1$ соотносятся следующим образом: $\mathbf{y}_j = \mathbf{x}_j + \text{round}(\mathbf{P}_0\mathbf{x}_i)$, $\mathbf{y}_i = \mathbf{x}_j + \text{round}(\mathbf{P}_1\mathbf{y}_j)$; $\mathbf{z}_i = \mathbf{y}_i - \text{round}(\mathbf{P}_1\mathbf{y}_j) = \mathbf{x}_i$, $\mathbf{z}_j = \mathbf{y}_j - \text{round}(\mathbf{P}_0\mathbf{y}_i) = \mathbf{x}_j$, где $\text{round}(\cdot)$ – оператор округления. В данном случае матрицы преобразования и их инверсные матрицы определяются как:

$$\begin{bmatrix}\mathbf{y}_i\\ \mathbf{y}_j\end{bmatrix} = \mathbf{T}\begin{bmatrix}\mathbf{x}_i\\ \mathbf{x}_j\end{bmatrix}, \begin{bmatrix}\mathbf{z}_i\\ \mathbf{z}_j\end{bmatrix} = \mathbf{T}^{-1}\begin{bmatrix}\mathbf{y}_i\\ \mathbf{y}_j\end{bmatrix} = \begin{bmatrix}\mathbf{x}_i\\ \mathbf{x}_j\end{bmatrix},$$

где $\mathbf{T} = \mathbf{P}_U\mathbf{P}_L, \mathbf{T}^{-1} = \mathbf{P}_L^{-1}\mathbf{P}_U^{-1}$,

$$\mathbf{P}_U = \begin{bmatrix}\mathbf{I} & \mathbf{P}_1\\ \mathbf{0} & \mathbf{I}\end{bmatrix}, \mathbf{P}_U^{-1} = \begin{bmatrix}\mathbf{I} & -\mathbf{P}_1\\ \mathbf{0} & \mathbf{I}\end{bmatrix}, \quad \mathbf{P}_L = \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{P}_0 & \mathbf{I}\end{bmatrix}, \mathbf{P}_L^{-1} = \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ -\mathbf{P}_0 & \mathbf{I}\end{bmatrix}.$$

Таким образом, следуя философии блочной лестничной схемной параметризации, матрицу оператора левого умножения кватернионов $\mathbf{M}^+(q)$ можно представить в виде блочной матрицы:

$$\mathbf{M}^+(q) = \begin{bmatrix}\mathbf{C}(q) & -\mathbf{S}(q)\\ \mathbf{S}(q) & \mathbf{C}(q)\end{bmatrix}, \ \mathbf{C}(q) = \begin{bmatrix}q_1 & -q_2\\ q_2 & q_1\end{bmatrix}, \ \mathbf{S}(q) = \begin{bmatrix}q_3 & q_4\\ q_4 & -q_3\end{bmatrix}.$$

Далее на основе известной лестничной схемной факторизации двумерной матрицы вращения

$$\mathbf{R}_2(\varphi)=\begin{bmatrix}\cos\varphi & -\sin\varphi\\ \sin\varphi & \cos\varphi\end{bmatrix}=\begin{bmatrix}1 & \cos\varphi-1/\sin\varphi\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0\\ \sin\varphi & 1\end{bmatrix}\begin{bmatrix}1 & \cos\varphi-1/\sin\varphi\\ 0 & 1\end{bmatrix}$$

можно получить следующую факторизацию матрицы умножения кватернионов:

$$\mathbf{M}^{+}(q)=\underbrace{\begin{bmatrix}\mathbf{I}_2 & \mathbf{F}(q)\\ 0 & \mathbf{I}_2\end{bmatrix}}_{\mathbf{U}(q)}\underbrace{\begin{bmatrix}\mathbf{I}_2 & 0\\ \mathbf{G}(q) & \mathbf{I}_2\end{bmatrix}}_{\mathbf{L}(q)}\underbrace{\begin{bmatrix}\mathbf{I}_2 & \mathbf{H}(q)\\ 0 & \mathbf{I}_2\end{bmatrix}}_{\mathbf{V}(q)}. \tag{4.1}$$

Для заданного коэффициента q и матрицы умножения $\mathbf{M}^{+}(q)$ определяется набор матричных выражений, которые могут быть решены однозначно для $\mathbf{F}(q)$, $\mathbf{G}(q)$, и $\mathbf{H}(q)$, при условии, что матрица $\mathbf{S}(q)$ является несингулярной, т. е. ненулевой:

$$\mathbf{F}(q)=(\mathbf{C}(q)-\mathbf{I}_2)\mathbf{S}(q)^{-1},\ \mathbf{G}(q)=\mathbf{S}(q),\ \mathbf{H}(q)=\mathbf{S}(q)^{-1}(\mathbf{C}(q)-\mathbf{I}_2). \tag{4.2}$$

Элементы данных матриц являются вещественными коэффициентами лестничной схемной параметризации.

В случае если в прямом преобразовании Q-ПУБФ используется умножение на нормированный кватернион $|q|=1$, то обратное преобразование основывается на умножении на сопряженный кватернион $\overline{q}$. Из равенства $\mathbf{M}^{\pm}(\overline{q})=\mathbf{M}^{\mp}(q)^T$ следует, что $\mathbf{M}^{+}(\overline{q})=\mathbf{V}(q)^T\mathbf{L}(q)^T\mathbf{U}(q)^T$, но здесь, в отличие от факторизации (4.1), нижние треугольные матрицы заменяются на верхние треугольные матрицы и наоборот. Однако, принимая во внимание, что $\mathbf{M}^{+}(\overline{q})=-\mathbf{P}\mathbf{M}^{+}(\overline{q})\mathbf{P}$, где $\mathbf{P}=\mathbf{\Gamma}_4\mathbf{J}_4$ и $\mathbf{P}^2=-\mathbf{I}_4$, факторизация оператора умножения на сопряженный кватернион имеет вид:

$$\mathbf{M}^{+}(\overline{q})=\underbrace{(-\mathbf{P})\mathbf{V}(q)^T\mathbf{P}}_{\mathbf{U}(\overline{q})}\underbrace{(-\mathbf{P})\mathbf{L}(q)^T\mathbf{P}}_{\mathbf{L}(\overline{q})}\underbrace{(-\mathbf{P})\mathbf{U}(q)^T\mathbf{P}}_{\mathbf{V}(\overline{q})}, \tag{4.3}$$

которая упрощается к

$$\mathbf{M}^{+}(\overline{q}) = \pm \underbrace{\begin{bmatrix} \mathbf{I}_2 & -\mathbf{H}(q) \\ 0 & \mathbf{I}_2 \end{bmatrix}}_{\mathbf{U}(\overline{q})} \underbrace{\begin{bmatrix} \mathbf{I}_2 & 0 \\ -\mathbf{G}(q) & \mathbf{I}_2 \end{bmatrix}}_{\mathbf{L}(\overline{q})} \underbrace{\begin{bmatrix} \mathbf{I}_2 & -\mathbf{F}(q) \\ 0 & \mathbf{I}_2 \end{bmatrix}}_{\mathbf{V}(\overline{q})}. \tag{4.4}$$

Структура обратимого оператора умножения кватернионов на основе блочной лестничной схемной параметризации показана на рисунке 4.2. Здесь белые кружки обозначают выходы, результат которых округлен.

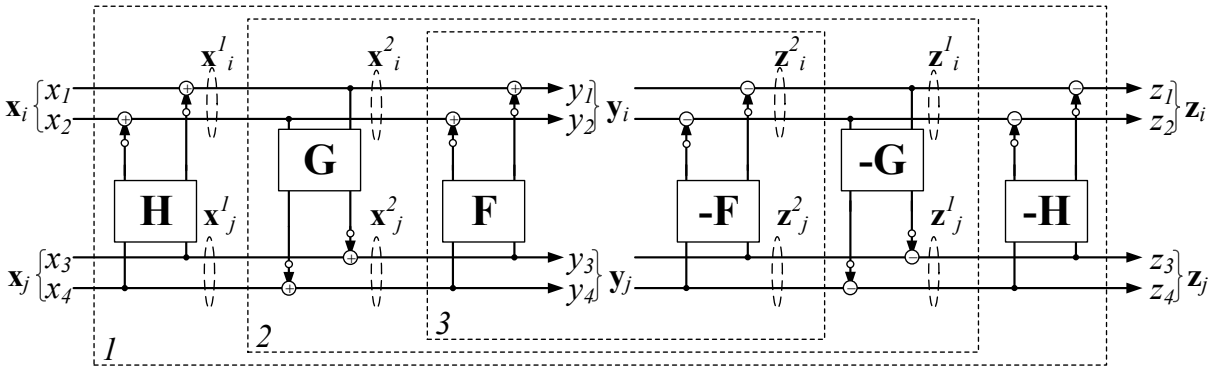


Рисунок 4.2 – Структура обратимого оператора умножения кватернионов на основе блочной лестничной схемной параметризации

В результате прямого преобразования входной вектор $\mathbf{x} = \left[x_i x_j \right]^T$ проходит три стадии обработки для формирования результата преобразования (умножения вектора $\mathbf{x}$ на коэффициент q) $\mathbf{y} = \left[y_i y_j \right]^T$:

$$\mathbf{x}_1 = \begin{bmatrix} \mathbf{I}_2 & \mathbf{H} \\ 0 & \mathbf{I}_2 \end{bmatrix} \mathbf{x} = \begin{bmatrix} x_i + \text{round}[\mathbf{H} x_j] \\ x_j \end{bmatrix} = \begin{bmatrix} x_i^1 \\ x_j^1 \end{bmatrix};$$

$$\mathbf{x}_2 = \begin{bmatrix} \mathbf{I}_2 & 0 \\ \mathbf{G} & \mathbf{I}_2 \end{bmatrix} \mathbf{x}_1 = \begin{bmatrix} x_i^1 \\ \text{round}[\mathbf{G} x_i^1] + x_j \end{bmatrix} = \begin{bmatrix} x_i^2 \\ x_j^2 \end{bmatrix};$$

$$\mathbf{y} = \begin{bmatrix} \mathbf{I}_2 & \mathbf{F} \\ 0 & \mathbf{I}_2 \end{bmatrix} \mathbf{x}_2 = \begin{bmatrix} x_i^1 + \text{round}[\mathbf{F} x_j^2] \\ x_j^2 \end{bmatrix} = \begin{bmatrix} y_i \\ y_j \end{bmatrix}.$$

Выходной результат обратного преобразования $\mathbf{z} = \left[z_i z_j \right]^T$ получается также после трех ступеней обработки результата прямого преобразования $\mathbf{y} = \left[y_i y_j \right]^T$:

$$\mathbf{z}_2 = \begin{bmatrix} \mathbf{I}_2 & -\mathbf{F} \\ 0 & \mathbf{I}_2 \end{bmatrix} \mathbf{y} = \begin{bmatrix} x_i^1 + \text{round}[\mathbf{F}x_j^2] - \text{round}[\mathbf{F}x_j^2] \\ x_j^2 \end{bmatrix} = \begin{bmatrix} x_i^1 \\ x_j^2 \end{bmatrix} = \mathbf{x}_2;$$

$$\mathbf{z}_1 = \begin{bmatrix} \mathbf{I}_2 & 0 \\ -\mathbf{G} & \mathbf{I}_2 \end{bmatrix} \mathbf{z}_2 = \begin{bmatrix} x_i^1 \\ -\text{round}[\mathbf{G}x_i^1] + \text{round}[\mathbf{G}x_i^1] + x_j \end{bmatrix} = \begin{bmatrix} x_i^1 \\ x_j \end{bmatrix} = \mathbf{x}_1;$$

$$\mathbf{z} = \begin{bmatrix} \mathbf{I}_2 & -\mathbf{H} \\ 0 & \mathbf{I}_2 \end{bmatrix} \mathbf{z}_1 = \begin{bmatrix} x_i + \text{round}[\mathbf{H}x_j] - \text{round}[\mathbf{H}x_j] \\ x_j \end{bmatrix} = \begin{bmatrix} x_i \\ x_j \end{bmatrix} = \mathbf{x}.$$

Таким образом, умножение на $1/q$, или на эквивалентный сопряженный кватернион $\overline{q}$, реализуется переупорядочиванием коэффициентов в обратном порядке и изменением их знака. Следует отметить, что округление ни лестничных коэффициентов, ни результата соответствующего умножения не влияет на обратимость преобразования. Следовательно, выполнение обратимого «целого к целому» отображения на основе оператора умножения возможно для арифметики с фиксированной запятой.

4.2 Прямая и обратная схемы умножителя кватернионов на основе блочной лестничной схемной параметризации со встроенным 2D CORDIC-модулем

Задание углов в пределах $-\pi \le \varphi < \pi$, $-\frac{\pi}{2} \le \psi \le \frac{\pi}{2}$, и $-\frac{\pi}{2} \le \chi \le \frac{\pi}{2}$ достаточно, чтобы описать любой кватернион заданного модуля $|q|$ и трех углов φ, ψ, χ. в полярной форме: $q = |q| e^{i\varphi} e^{j\psi} e^{k\chi}$, где $q_1 = |q| \cos\varphi$, $q_2 = |q| \sin\varphi \cos\psi$, $q_3 = |q| \sin\varphi \sin\psi \cos\chi$, $q_4 = |q| \sin\varphi \sin\psi \sin\chi$. Для $|q| = 1$ расчетные выражения блочной лестничной схемной параметризации (4.2) оператора умножения кватернионов могут быть представлены следующим образом:

$$
\begin{aligned}
&f_{11}(q) = -f_{22}(q) = [\cos\chi(\cos\varphi - 1) - \sin\varphi\cos\psi\sin\chi]/d,\\
&f_{12}(q) = f_{21}(q) = [\sin\chi(\cos\varphi - 1) + \sin\varphi\cos\psi\cos\chi]/d,\\
&g_{11}(q) = -g_{22}(q) = d\cos\chi,\\
&g_{12}(q) = g_{21}(q) = d\sin\chi,\\
&h_{11}(q) = -h_{22}(q) = [\cos\chi(\cos\varphi - 1) + \sin\varphi\cos\psi\sin\chi]/d,\\
&h_{12}(q) = h_{21}(q) = [\sin\chi(\cos\varphi - 1) - \sin\varphi\cos\psi\cos\chi]/d\ ,\\
&d = \sin\varphi\cos\psi,
\end{aligned}
\tag{4.5}
$$

Анализ (4.5) показывает, что в компонентах $\mathbf{F}(q)$, $\mathbf{G}(q)$ и $\mathbf{H}(q)$ факторизации матрицы умножения кватернионов $\mathbf{M}^{+}(q)$ (4.1) есть пары коэффициентов с одинаковым модулем, что не очевидно из (4.2) и не принималось во внимание, когда создавалась факторизация (4.1). Следовательно структуры матриц $\mathbf{F}(q)$, $\mathbf{G}(q)$ и $\mathbf{H}(q)$ имеют структуру, близкую к матрице $\mathbf{R}_2(\varphi)$ вращения Гивенса $\begin{bmatrix} s & c \\ c & -s \end{bmatrix} = \mathbf{J}_2 \begin{bmatrix} c & -s \\ s & c \end{bmatrix}$, где через c и s обозначены косинус и синус соответствующего угла. Например, матрица $\mathbf{F}(q)$ имеет следующий вид:

$$
\begin{bmatrix} f_{11}(q) & f_{12}(q) \\ f_{12}(q) & -f_{11}(q) \end{bmatrix} = \mathbf{J}_2 \begin{bmatrix} f_{12}(q) & -f_{11}(q) \\ f_{11}(q) & f_{12}(q) \end{bmatrix}.
$$

Таким образом, данные матрицы могут быть аппроксимированы с использованием 2D CORDIC-алгоритма, а зависимости (4.5) являются расчетными выражениями блочной лестничной схемной параметризации оператора умножения кватернионов, а также формируют матрицы вращения Гивенса для соответству-

ющего этапа разложения $\mathbf{U}(q)$, $\mathbf{L}(q)$, $\mathbf{V}(q)$. Прямая и обратная схемы умножителя кватернионов на основе блочной лестничной схемной параметризации со встроенным 2D CORDIC-модулем приведены на рисунке 4.3.

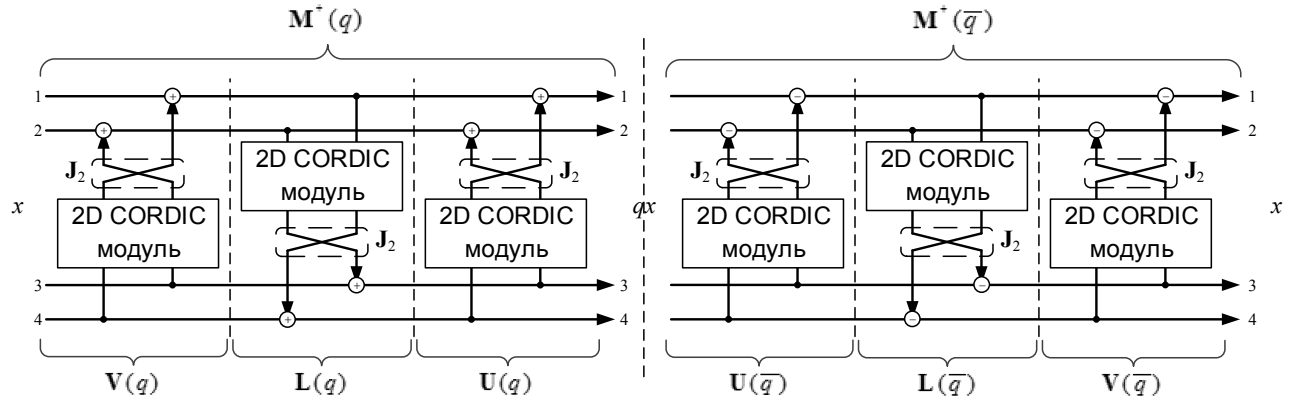


Рисунок 4.3 – Прямая и обратная схемы умножителя кватернионов на основе блочной лестничной параметризации со встроенным 2D CORDIC-модулем

Динамический диапазон данной схемы умножителя кватернионов ограничен $-1\ldots1$, что обусловлено форматом фиксированной запятой. Однако, коэффициенты, полученные с помощью расчетных выражения блочной лестничной схемной параметризации (4.5) оператора умножения кватернионов, могут иметь абсолютные значения больше единицы. Приведение параметров умножителя к требуемому динамическому диапазону может быть достигнуто заменой умножения кватерниона q на кватернион $\tilde{q}$, который представляет собой версию кватерниона q с переставленными и/или измененными знаками компонент. Матрицы оператора умножения кватернионов $\mathbf{M}^{\pm}(q)$ и $\mathbf{M}^{\pm}(\tilde{q})$ будут соотносится согласно следующему выражению:

$$\mathbf{M}^{\pm}(q) = \mathbf{P}_{POST}\mathbf{M}^{\pm}(\tilde{q})\mathbf{P}_{PRE}\ , \tag{4.6}$$

где матрицы $\mathbf{P}_{PRE}$ и $\mathbf{P}_{POST}$ являются матрицами перестановок исходного кватерниона q и кватерниона $\tilde{q}$ с переставленными и/или измененными знаками компонент кватерниона q. На практике это соответствует лишь выполнению пре- и

постобработки входного операнда $\mathbf{x}$ и выходного результата соответственно: $qx = \mathbf{M}^{\pm}(q)\mathbf{x} = \mathbf{P}_{POST}\mathbf{M}^{\pm}(\tilde{q})\mathbf{P}_{PRE}\,\mathbf{x}$.

Таким образом, гибридная схема CORDIC-лестничной параметризации позволяет интегрировать 2D CONST CORDIC-алгоритм «внутрь» лестничной схемы умножителя, заменив действительные умножения на микровращения CORDIC-алгоритма: сложение и сдвиг, а также получить перфективное преобразование и в аппаратной реализации. Неотъемлемым свойством блочных лестничных схем является то, что все результаты между этапами факторизации могут быть вычислены сразу, без использования вспомогательной памяти. Небольшая разница в схемах для прямого и обратного умножения (умножения на сопряженный кватернион) позволяет проектировать универсальный умножитель. Разница лишь в знаке для коэффициентов прямого и обратного преобразования, что позволяет иметь одну схему с переключаемыми режимами. Другое преимущество предложенной схемы умножителя кватернионов заключается в том, что число операций округления уменьшилось почти в два раза по сравнению со структурой умножителя на основе лестничной схемной параметризации.

Процесс оптимизации параметров встроенного 2D CONST CORDIC выполняется в три этапа:

1) для кватерниона q рассматриваются все модификации гиперкомплексного числа, которые потенциально определяют вычислительную схему умножителя, в которой все коэффициенты лестничной схемы находятся в диапазоне $-1\ldots1$;

2) для каждого такого модифицированного кватерниона $\tilde{q}$ осуществляется формирование матриц $\mathbf{P}_{pre}^{CORDIC}$ и $\mathbf{P}_{post}^{CORDIC}$ (2.4)–(2.5), расчет параметров и оценивается число итераций 2D CONST CORDIC-алгоритма (число микровращений итерационного процесса и количество итераций процесса масштабирования), гарантирующее заданную точность операции умножения на кватернион-константу;

3) выбирается соответствующий кандидат из множества модифицированных кватернионов $\tilde{q}$, для которого параметры схемы наиболее близко согласуются с принятой платформой реализации, например, минимальное число итераций, что обеспечивает максимальную скорость выполнения операции умножения.

Например, для кватерниона $q = q_1 + q_2 i + q_3 j + q_4 k = 1/\sqrt{30}\,(4 - i + 3j - 2k)$ процесс настройки параметров умножителя на основе 2D CONST CORDIC-модуля выглядит следующим образом. На первом этапе для обеспечения заданного динамического диапазона из 24 возможных модификаций исходного кватерниона $q = q_1 + q_2 i + q_3 j + q_4 k$ только у 7 кватернионов $\tilde{q}$, представляющих собой версии кватерниона q с переставленными и/или измененными знаками компонент, коэффициенты блочной лестничной схемной параметризации (4.5) попадают в динамический диапазон $-1\ldots1$. У модифицированного кватерниона $\tilde{q} = q_3 + q_2 i + q_1 j + q_4 k$ коэффициенты блочной лестничной факторизации (4.5) следующие: $f_{11}(q) = -f_{22}(q) = -0{,}395$, $f_{12}(q) = f_{21}(q) = 0{,}488$, $g_{11}(q) = -g_{22}(q) = 0{,}730$, $g_{12}(q) = g_{21}(q) = 0{,}365$, $h_{11}(q) = -h_{22}(q) = -0{,}595$, $h_{12}(q) = h_{21}(q) = 0{,}048$. Далее, на втором и третьем этапах процесса оптимизации параметров встроенного 2D CONST CORDIC расчет числа итераций (алгоритмы 1 и 2) 2D CORDIC-модуля для каждого этапа разложения $\mathbf{U}(q)$, $\mathbf{L}(q)$, $\mathbf{V}(q)$ схемы умножителя кватернионов (см. рисунок 4.3) показал, что у данного модифицированного кватерниона $\tilde{q} = q_3 + q_2 i + q_1 j + q_4 k$ для точности ε_φ и ε_K порядка 10^{-5} требуется минимальное число микро вращений итерационного процесса 7 ($M_u = 3$, $M_L = 1$, $M_V = 3$) и итераций процесса масштабирования 17 ($L_u = 7$, $L_L = 5$, $L_u = 5$). В таблице 2 для кватерниона $\tilde{q}$ приведены параметры управления $\tau(i)$ и $\sigma(i)$, а также L и $s(l)$ процесса масштабирования для каждой компоненты разложения $\mathbf{U}(q)$, $\mathbf{L}(q)$, $\mathbf{V}(q)$. Для реализации умножения на ква-

тернион q через умножение на кватернион $\tilde{q}$ должна быть выполнена следующая пре- и постобработка (4.6) на соответствующие матрицы перестановок данных:

$$\mathbf{P}_{PRE} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}, \quad \mathbf{P}_{POST} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}.$$

Таблица 4.1 – Параметры управления 2D CONST CORDIC-модулей схемы умножения кватернионов на основе блочной лестничной параметризации

Параметр управления	Компоненты разложения $\mathbf{U}$, $\mathbf{L}$, $\mathbf{V}$		
	Компонента $\mathbf{U}$: $M=3$ и $L=7$	Компонента $\mathbf{L}$: $M=1$ и $L=5$	Компонента $\mathbf{V}$: $M=3$ и $M_v=5$
$\sigma(0)$	1	-1	1
$\tau(0)$	0	1	4
$\sigma(1)$	-1	–	1
$\tau(1)$	4		6
$\sigma(2)$	1		1
$\tau(2)$	11		9
$s(0)$	1	2	2
$s(1)$	3	6	3
$s(2)$	5	7	4
$s(3)$	8	9	6
$s(4)$	10	10	6
$s(5)$	12	–	–
$s(6)$	13		
$\mathbf{P}_{pre}^{CORDIC}$	$-\mathbf{\Gamma}_2$	$\mathbf{I}_2$	$-\mathbf{I}_2$
$\mathbf{P}_{post}^{CORDIC}$	$\mathbf{J}_2$	$\mathbf{I}_2$	$\mathbf{I}_2$

Аппаратные затраты умножителя кватернионов 4D CONST Q-CORDIC (см. рисунок 3.2) приблизительно на 50 % меньше, чем у умножителя кватернионов с блочной лестничной структурой на основе 2D CONST CORDIC (см. рису-

нок 4.3). Это обусловлено дополнительными аппаратными затратами на встраивание модуля 2D CONST CORDIC в схему умножителя для обеспечения динамического диапазона обработки данных в пределах $-1\ldots1$ и вращения на произвольный угол (реализация матриц перестановок (4.6) и (2.5)).

5 ЛАБОРАТОРНЫЙ ПРАКРИКУМ

5.1.1 Лабораторная работа №1. Схема устройства 2D CORDIC алгоритма

Цель работы: разработать схему устройства CORDIC для чисел с фиксированной запятой в дополнительном коде, исследовать его работу средствами САПР Xilinx.

Вариант задания:

- разрядность (бит): 8; 12; 14; 16;
- вращение вектора на угол (градусы): 12; 24; 28; 32; 36;

Порядок выполнения работы:

Часть 1. Разработать схему устройства 2D CORDIC (стандартный)

1) рассчитать погрешность операции;

2) синтезировать схему арифметического устройства для выполнения одной итерации алгоритма CORDIC (при синтезе данной схемы использовать язык VHDL);

3) исследовать работу устройства средствами САПР ПЛИС Xilinx ISim;

4) определить время критического пути;

5) оценить производительность устройства;

Часть 2. Разработать схему устройства 2D CONST CORDIC:

1) рассчитать и реализовать схему устройства 2D CONST CORDIC;

2) исследовать работу устройства.

Содержание отчета

Протокол работы должен содержать схему арифметического устройства и схему синхронизации CORDIC, временные диаграммы работы устройства на од-

ной итерации и при выполнении преобразований в целом в двух основных режимах. При реализации устройства средствами языка VHDL приводится текст программы.

5.2 Лабораторная работа №2. Схема конвейерного процессора 2D CORDIC-алгоритма

Цель работы: разработать VHDL-описание конвейерного процессора:

1) 2D CORDIC-алгоритм с разреженными итерациями;
2) 2D CONST CORDIC-алгоритм,

исследовать его работу средствами САПР Xilinx ISim.

Порядок выполнения работы:

- разработать VHDL-описание;
- получить временные диаграммы его работы;
- определить время латенции;
- оценить объем занимаемых ресурсов и быстродействие;
- сравнить производительность конвейерного процессора с предыдущим устройством.

Содержание отчета

Протокол работы должен содержать схему арифметического устройства и схему синхронизации CORDIC, временные диаграммы работы устройства на одной итерации и при выполнении преобразований в целом в двух основных режимах. При реализации устройства средствами языка VHDL приводится текст описания.

5.3 Лабораторная работа №3. Схема процессора 4D CORDIC-алгоритма с разреженными итерациями (умножитель кватернионов)

Цель работы: разработать схему устройства 4D CORDIC для чисел с фиксированной запятой в дополнительном коде, исследовать его работу средствами САПР Xilinx ISim.

Порядок выполнения работы:

1) рассчитать погрешность операции;

2) синтезировать схему арифметического устройства для выполнения одной итерации алгоритма CORDIC. При синтезе данной схемы и всех последующих возможно использование языка VHDL;

3) исследовать работу устройства средствами САПР Xilinx ISim;

4) определить время критического пути;

5) оценить производительность устройства;

Содержание отчета

Протокол работы должен содержать схему арифметического устройства и схему синхронизации 4D CORDIC, временные диаграммы работы устройства на одной итерации и при выполнении преобразований в целом в двух основных режимах. При реализации устройства средствами языка VHDL приводится текст описания.

5.4 Лабораторная работа №4. Целочисленный умножитель на кватернион константу на основе встроенного 2D CONST CORDIC-процессора

Цель работы: разработать схему устройства целочисленного умножителя кватернионов для чисел с фиксированной запятой в дополнительном коде, исследовать его работу средствами САПР Xilinx ISim.

Вариант задания:

- разрядность, (бит): 8; 12; 14; 16;

- кватернион константа: $q = [4, -1, 3, -2] / \sqrt{30}$;

Порядок выполнения работы:

Часть 1. Разработать схему устройства 2D CONST CORDIC:

1) рассчитать погрешность операции;

2) синтезировать схему арифметического устройства для выполнения одной итерации алгоритма CORDIC. При синтезе данной схемы и всех последующих возможно использование языка VHDL;

3) исследовать работу устройства средствами САПР Xilinx ISim;

4) определить время критического пути;

5) оценить производительность устройства;

Часть 2. Разработать схему устройства целочисленного умножителя кватернионов на базе 2D CONST CORDIC

1) исследовать работу устройства средствами САПР Xilinx ISim;

2) определить время критического пути;

3) оценить производительность устройства.

Содержание отчета

Протокол работы должен содержать схему арифметического устройства и схему синхронизации умножителя, временные диаграммы работы устройства. При реализации устройства средствами языка VHDL приводится текст описания.

ПРИЛОЖЕНИЕ А
(обязательное)
Вращение вектора на плоскости

Изучение CORDIC-алгоритма можно начать с задачи вращения вектора (либо точки) на заданный угол α (рис.А.1). Как известно, вектор в двумерном пространстве задается парой координат (x_1, y_1). Вращение вектора на угол α можно записать в матричном виде следующим образом:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \mathbf{R}(\alpha) \times \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}, \qquad (A.1)$$

где $\mathbf{R}(\alpha)$ – матриц вращения, которая задается как

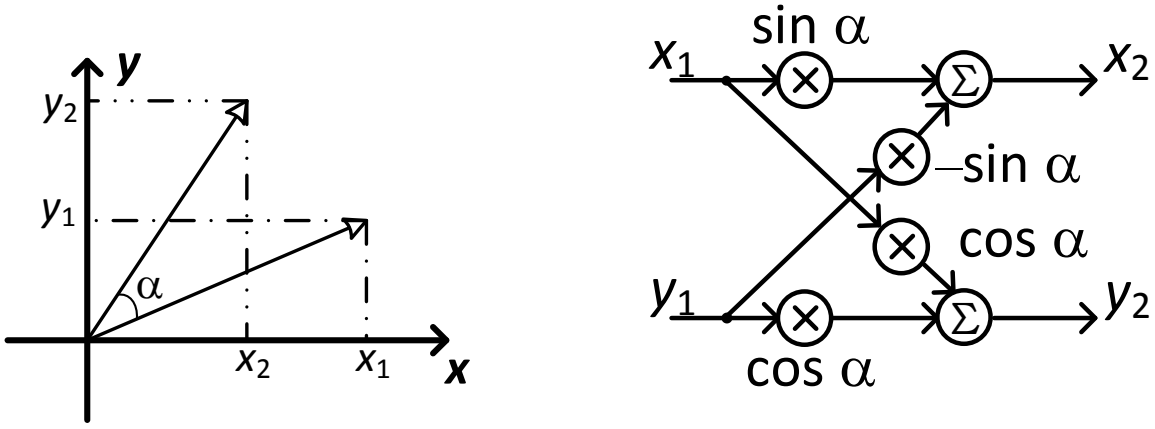
$$\mathbf{R}(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}. \qquad (A.2)$$

Прямая реализация выражения (А.1) потребует выполнения четырех операций умножения и двух операций сложения. В то время как использование CORDIC-алгоритма позволяет выполнить операцию вращения вектора без использования операций умножения. Преобразуем матрицу вращения следующим образом:

$$\mathbf{R}(\alpha) = \cos\alpha \times \begin{bmatrix} 1 & -\mathrm{tg}\alpha \\ \mathrm{tg}\alpha & 1 \end{bmatrix}. \qquad (A.3)$$

Далее, используя соотношение $\cos\alpha = 1/\sqrt{1+(\mathrm{tg}\alpha)^2}$ получим:

$$\mathbf{R}(\alpha) = \frac{1}{\sqrt{1+(\mathrm{tg}\alpha)^2}} \times \begin{pmatrix} 1 & -\mathrm{tg}\alpha \\ \mathrm{tg}\alpha & 1 \end{pmatrix}. \qquad (A.4)$$



а б

Рисунок. А.1. – (а) Вращение вектора на плоскости; (б) прямая реализация вращения

Отметим одно свойство матрицы вращения, которое понадобится нам для дальнейших выкладок:

$$\mathbf{R}(\alpha+\beta)=\mathbf{R}(\alpha)\times\mathbf{R}(\beta). \qquad (A.5)$$

Основная идея CORDIC-алгоритма заключается в том, чтобы исходный угол вращения α представить в виде линейной комбинации заранее заданных (базовых) углов θ_i:

$$\alpha\approx\sum_{i=0}^{N_A-1}\mu(i)\theta_i, \qquad (A.6)$$

где N_A – количество используемых базовых углов; $\mu(i)\in\{1,-1\}$ является последовательностью, определяющей вращение. Базовые углы θ_i задаются следующим выражением:

$$\theta_i\triangleq\operatorname{arctg}\left(2^{-i}\right). \qquad (A.7)$$

Далее, используя (А.5) и (А.6), получаем:

$$\mathbf{R}(\alpha) \approx \mathbf{R}\left(\sum_{i=0}^{N_A-1}\mu(i)\theta_i\right) =$$
$$= \mathbf{R}(\mu(0)\theta_0) \times \mathbf{R}(\mu(1)\theta_1) \ldots \mathbf{R}(\mu(N_A-1)\theta_{N_A-1}) = \quad \text{(A.8)}$$
$$= \prod_{i=0}^{N_A-1}\mathbf{R}(\mu(i)\theta_i).$$

Используя (A.8) выражение (A.4) переписывается в виде:

$$\mathbf{R}(\alpha) \approx \prod_{i=0}^{N_A-1}\mathbf{R}(\mu(i)\theta_i) = \prod_{i=0}^{N_A-1}\frac{1}{\sqrt{1+2^{-2i}}} \times \begin{bmatrix} 1 & -\mu(i)2^{-i} \\ \mu(i)2^{-i} & 1 \end{bmatrix}. \quad \text{(A.9)}$$

При выводе последнего выражения учитывалось, что $(\mu(i))^2 = 1$ и то, что $\mathrm{tg}(\mu(i)\theta_i) = \mu(i)2^{-i}$. Перепишем (A.9) в следующем виде:

$$\mathbf{R}(\alpha) \approx K_{N_A} \times \prod_{i=0}^{N_A-1}\begin{bmatrix} 1 & -\mu(i)2^{-i} \\ \mu(i)2^{-i} & 1 \end{bmatrix}, \quad \text{(A.10)}$$

где

$$K_{N_A} = \prod_{i=0}^{N_A-1}\frac{1}{\sqrt{1+2^{-2i}}}. \quad \text{(A.11)}$$

Таким образом, задача вращения вектора сводится к выполнению последовательных операций сдвига и сложения, с последующим масштабированием выходных данных на коэффициент K_{N_A}. Важно отметить, что как только выбрано множество базовых углов, коэффициент масштабирования K_{N_A} оказывается фиксированным и постоянным для всех вращений. Ниже приведен пример использования CORDIC-алгоритма.

Пример. Выполнить вращение вектора на угол $\alpha = \frac{\pi}{6} = 0,523598$ радиан, используя CORDIC алгоритм ($N_A = 8$). Вначале составим таблицу базовых углов:

Индекс	Базовый угол	Значение в радианах
$i=0$	$\theta_0 = \mathrm{arctg}\left(2^{-0}\right)$	$0,785398$
$i=1$	$\theta_1 = \mathrm{arctg}\left(2^{-1}\right)$	$0,463647$
$i=2$	$\theta_2 = \mathrm{arctg}\left(2^{-2}\right)$	$0,244978$
$i=3$	$\theta_3 = \mathrm{arctg}\left(2^{-3}\right)$	$0,124354$
$i=4$	$\theta_0 = \mathrm{arctg}\left(2^{-4}\right)$	$0,062419$
$i=5$	$\theta_0 = \mathrm{arctg}\left(2^{-5}\right)$	$0,031240$
$i=6$	$\theta_0 = \mathrm{arctg}\left(2^{-6}\right)$	$0,015623$
$i=7$	$\theta_0 = \mathrm{arctg}\left(2^{-7}\right)$	$0,007812$

Также для $\mathrm{N_A} = 8$ по формуле (А.11) рассчитаем масштабирующий множитель:

$$K_{N_A} = \prod_{i=0}^{7} \frac{1}{\sqrt{1+2^{-2i}}} = 0,607259.$$

Далее определим последовательность $\mu(\mathrm{i})$, определяющую вращение. $\alpha \approx \theta_0 - \theta_1 + \theta_2 - \theta_3 + \theta_4 + \theta_5 - \theta_6 + \theta_7 = 0,528221$,

Таким образом, $\mu(\mathrm{i}) = \left[1 \quad -1 \quad 1 \quad -1 \quad 1 \quad 1 \quad -1 \quad 1\right]$, при этом ошибка аппроксимации угла α равна:

$$\varepsilon = \alpha - \sum_{i=0}^{7} \mu(i)\theta_i \approx -0,004623 \text{ рад.}$$

Ниже показана последовательность микровращений, которая имеет место при использовании CORDIC-алгоритма. Путь исходный вектор имеет координаты $(\mathrm{x},\mathrm{y}) = (1;0)$.

Номер итерации	**Микровращение**
$i=0$	$\begin{bmatrix}1\\1\end{bmatrix}=\begin{bmatrix}1 & -2^{-0}\\2^{-0} & 1\end{bmatrix}\times\begin{bmatrix}1\\0\end{bmatrix}$
$i=1$	$\begin{bmatrix}3/2\\1/2\end{bmatrix}=\begin{bmatrix}1 & 2^{-1}\\-2^{-1} & 1\end{bmatrix}\times\begin{bmatrix}1\\1\end{bmatrix}$
$i=2$	$\begin{bmatrix}11/2^3\\7/2^3\end{bmatrix}=\begin{bmatrix}1 & -2^{-2}\\2^{-2} & 1\end{bmatrix}\times\begin{bmatrix}3/2\\1/2\end{bmatrix}$
$i=3$	$\begin{bmatrix}95/2^6\\45/2^6\end{bmatrix}=\begin{bmatrix}1 & 2^{-3}\\-2^{-3} & 1\end{bmatrix}\times\begin{bmatrix}11/2^3\\7/2^3\end{bmatrix}$
$i=4$	$\begin{bmatrix}1475/2^{10}\\815/2^{10}\end{bmatrix}=\begin{bmatrix}1 & -2^{-4}\\2^{-4} & 1\end{bmatrix}\times\begin{bmatrix}95/2^6\\45/2^6\end{bmatrix}$
$i=5$	$\begin{bmatrix}46385/2^{15}\\27555/2^{15}\end{bmatrix}=\begin{bmatrix}1 & -2^{-5}\\2^{-5} & 1\end{bmatrix}\times\begin{bmatrix}1475/2^{10}\\815/2^{10}\end{bmatrix}$
$i=6$	$\begin{bmatrix}2971395/2^{21}\\1717135/2^{21}\end{bmatrix}=\begin{bmatrix}1 & 2^{-6}\\-2^{-6} & 1\end{bmatrix}\times\begin{bmatrix}46385/2^{15}\\27555/2^{15}\end{bmatrix}$
$i=7$	$\begin{bmatrix}378621425/2^{28}\\222764675/2^{28}\end{bmatrix}=\begin{bmatrix}1 & -2^{-7}\\2^{-7} & 1\end{bmatrix}\times\begin{bmatrix}2971395/2^{21}\\1717135/2^{21}\end{bmatrix}$
Масштабирование	$\begin{bmatrix}0,856524\\0,503942\end{bmatrix}=0,607259\times\begin{bmatrix}1,410475\\0,829863\end{bmatrix}$

Реализация умножения на константу (масштабирование) можно выполнить с использованием четырех сложений, если представить коэффициент масштабирования как

$$0,607259\approx 2^{-1}+2^{-3}-2^{-6}-2^{-9}-2^{-13}.$$

Таким образом, координатами вектора $(1;0)$, повернутого на $\frac{\pi}{6}$ рад являются $(0,856524;0,503942)$, который были найдены посредством CORDIC-алгоритма.

ПРИЛОЖЕНИЕ Б
(обязательное)
Вычисление прикладных функций на базе CORDIC-алгоритма

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

$$\sigma_i = \mathrm{sign}(z_i), \tag{Б.1}$$

$$z_{i+1} = z_i - \sigma_i \alpha_i. \tag{Б.2}$$

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:

$$\begin{aligned} &\sigma_i = -\mathrm{sign}(y_i), \\ &z_{i+1} = z_i - \sigma_i \alpha_i, \\ &i = 0, \ldots, n-1, z = 0, \alpha_i = \mathrm{arctg}\left(2^{-i}\right). \end{aligned} \tag{Б.3}$$

После выполнения n итераций получим:

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

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 «вращение». Таким образом, варьируя начальные значения и используя различные правила для определения значений управляющих операторов, можно вычислять основные тригонометрические функции, а также осуществлять преобразования координат.

Вычисление синуса и косинуса.

Выбирая в качестве начальных данных $x_0 = 1$, $x_0 = 0$, $z_0 = \varphi$, после выполнения n итераций в режиме операции «вращение», получим в результате следующие значения: $x_n = K_n \cdot \cos(\varphi)$, $y_n = K_n \cdot \sin(\varphi)$. 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 $x_0 = K_n^{-1}$.

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

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

Вычисление длины вектора. После выполнения n итераций в режиме «вектор» с взятыми в качестве начальных условий координатами вектора и $z_0 = 0$, получим $x_n = K_n$.

Вычисление обратных функций. Для нахождения обратных тригонометрических функций, таких, как арксинус и арккосинус, может быть использован режим операции «вращение» 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.

Операции умножения/деления. Определим следующие рекуррентные соотношения:

$$\begin{aligned} x_{i+1} &= x_i - 0 y_i \sigma_i 2^{-i} = x_i \\ y_{i+1} &= y_i + x_i \sigma_i 2^{-i} \\ z_{i+1} &= z_i - \sigma_i 2^{-i} \end{aligned} \qquad \text{(Б.5)}$$

Правила выбора управляющих операторов остаются такими же, как и в случае исходного CORDIC-алгоритма (Б.2), (Б.4). Ниже приводятся результаты вычислений в режиме операции «вращение» и в режиме операции «вектор» после выполнения n итераций.

$$x_n = x_0,\ y_n = y_0 + x_0 z_0,\ z_n = 0, \qquad \text{(Б.6)}$$

$$x_n = x_0,\ y_n = 0,\ z_n = z_0 - \frac{y_0}{x_0}. \qquad \text{(Б.7)}$$

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:

$$\begin{aligned} x_{i+1} &= x_i - m y_i \sigma_i 2^{-i} = x_i, \\ y_{i+1} &= y_i + x_i \sigma_i 2^{-i}, \\ z_{i+1} &= z_i - \sigma_i 2^{-i}. \end{aligned} \qquad \text{(Б.8)}$$

Здесь параметр m определяет систему, в которой действует алгоритм: $m=1$ – обычное вращение вектора на плоскости; $m=0$ – множительно-делительные операции (линейная система); $m=-1$ – вращение в гиперболических координатах. Вспомогательные величины α_i зависят от выбранной системы и определяются следующим образом: $\alpha_i=\arctan\left(2^{-i}\right)$ при $m=1$, $\alpha_i=2^{-i}$ при $m=0$, $\alpha_i=\operatorname{arcth}\left(2^{-i}\right)$ при $m=-1$. Правила выбора для управляющей последовательности операторов σ_i зависят от выбранного режима операций и определяются формулами (Б.2) либо (Б.4) соответственно.

ПРИЛОЖЕНИЕ В.
(обязательное)
Примеры VHDL-описания некоторых функциональных блоков

Для многих функциональных блоков программа-синтезатор Xilinx ISE имеет стандартные способы описания, которые рекомендуется использовать в процессе проектирования устройств на базе ПЛИС. Далее рассматриваются примеры наиболее часто встречающихся функциональных блоков.

Регистр с сигналом разрешения записи

Рассмотрим VHDL-описание регистра, изображенного на рисунке 3.

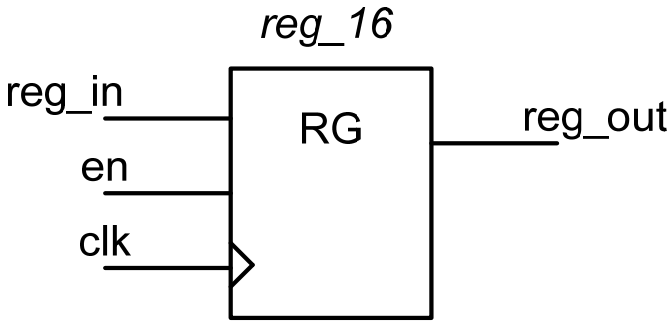


Рисунок В.1 – Регистр с сигналом разрешения записи *en*

```
library ieee;
use ieee.std_logic_1164.all;

entity reg_16 is
    port(reg_in : in std_logic_vector(15 downto 0);
         clk    : in std_logic;  -- тактовый сигнал
         en     : in std_logic;  -- сигнал разрешения записи
         reg_out: out std_logic_vector(15 downto 0));
end reg_16;

architecture arch of reg_16 is
signal tmp: std_logic_vector(15 downto 0);
begin

process (clk)
begin
```

```
    if (clk'event and clk ='1') then
        if (en='1') then
            tmp <= reg_in;
        end if;
    end if;
end process;

reg_out <= tmp;

end arch;
```

Сумматор/вычитатель

Ниже приведен код для описания сумматора/вычитателя.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_signed.all;

entity add_sub is
    Port ( a : in  std_logic_vector (15 downto 0);
           b : in  std_logic_vector (15 downto 0);
           vibor : in  std_logic;
           c : in  std_logic_vector (15 downto 0));
end add_sub;

architecture Behavioral of add_sub is

begin

process (a, b, vibor)
begin
   if vibor = '1' then
      c <= a + b;
   else
      c <= a - b;
```

```
    end if;
end process;
end Behavioral;
```

Реализация фильтра с бесконечной импульсной характеристикой второго порядка

Блок-схема звена второго порядка показана на рисунке В.2.

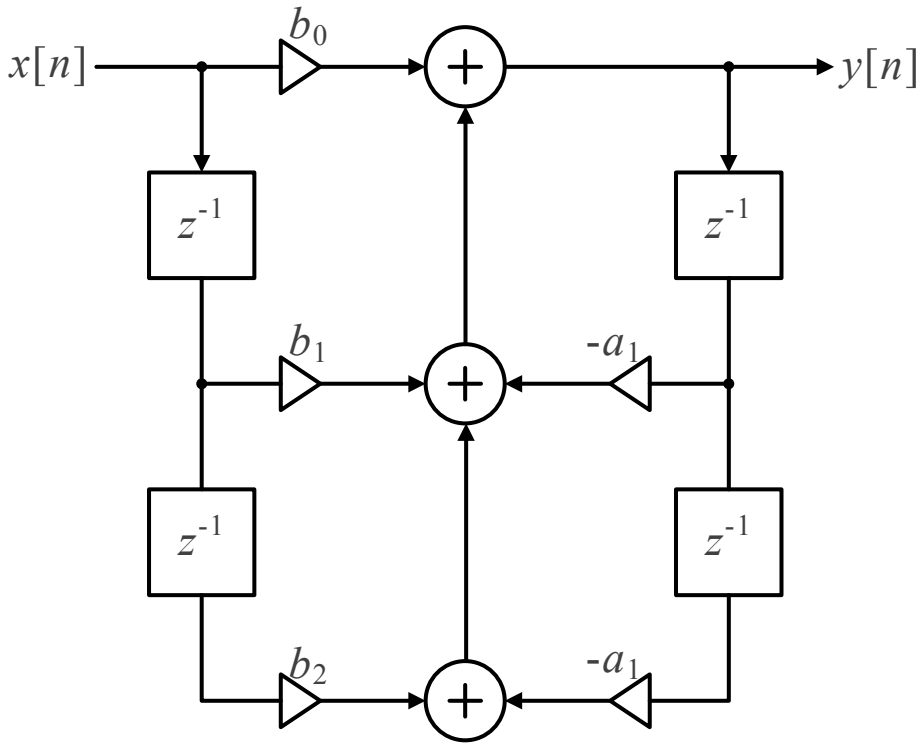


Рисунок В.2 – Звено второго порядка

На рисунке В.3 приведен возможный вариант реализации второго порядка.

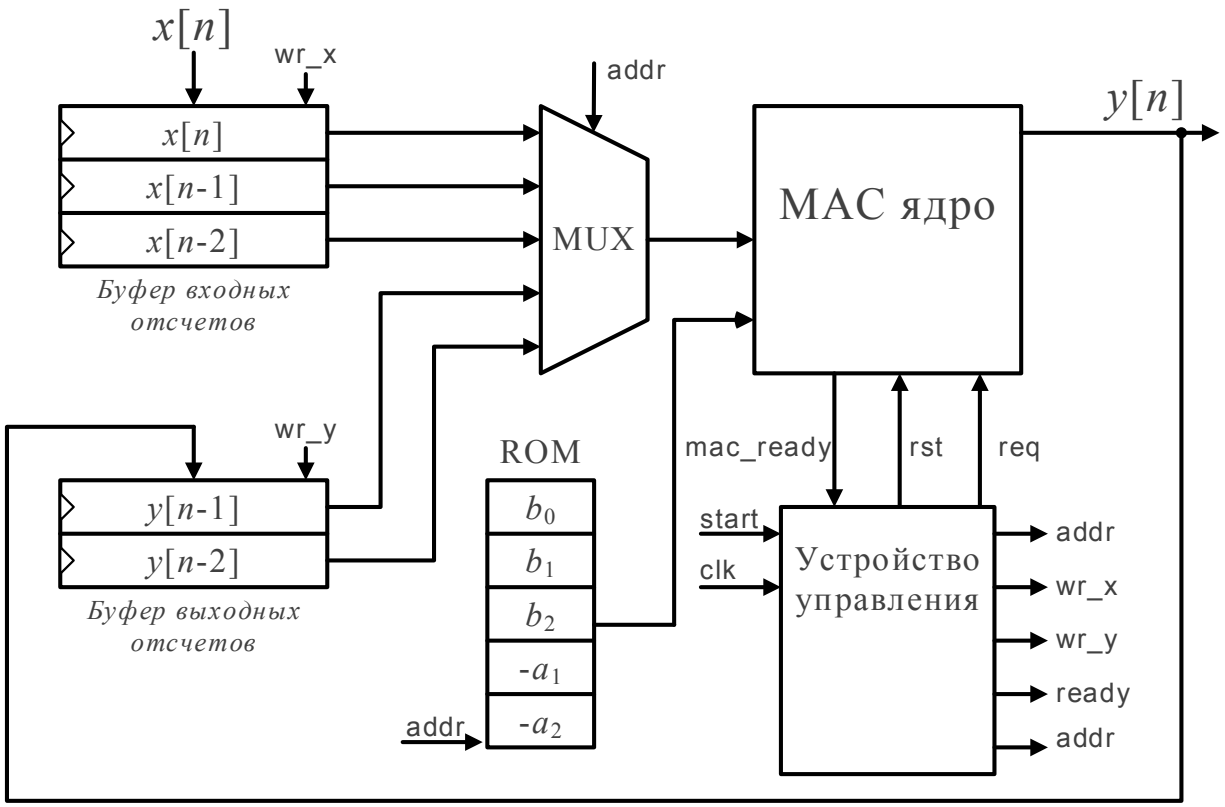


Рисунок В.3 – Функциональная схема реализации звена второго порядка

Схема начинает работу по входному сигналу `start`, поступающему на устройство управления (УУ). В свою очередь УУ формирует сигнал `wr_x` для записи нового отсчета $x[n]$ в буфер входных отсчетов, состоящий из трех регистров. Далее запускается процесс вычисление выходного отсчета $y[n]$. Вначале выполняется обнуления внутреннего регистра MAC-ядра сигналом `rst`. Далее сигнал `addr` устанавливается равным нулю, при этом из памяти (ROM) считывается коэффициент b_0, а через мультиплексор на второй вход MAC-ядра подается отсчет $x[n]$. Далее запускается процесс умножение сигналом `req`. По выполнении умножения MAC-ядро выставляет сигнал `mac_ready`, после чего сигнал `addr` устанавливается равным единицы, на вход MAC-ядра поступают b_1 и $x[n-1]$ и повторяется процесс умножения/накопления. После того, как выполнено умножение на все коэффициенты при помощи сигнала `wr_y` рассчитанное

значение $y[n]$ записывается в буфер выходных отсчетов, а все устройство переходит в режим ожидания прихода нового отсчета (который сопровождается сигналом `start`).

Ниже приводится MATLAB-модель, описывающая работу звена второго порядка.

```
[b,a] = butter(2,0.5);
[h, w]=freqz(b,a);
plot(w/pi,abs(h))
% x = 0.2*sin(2*pi*(0:127)*(61/128)) + cos(2*pi*(0:127)*(5/128));
x = [1 zeros(1,128)];
y = zeros(1,length(x));
reg_xnm1 = 0;
reg_xnm2 = 0;
reg_ynm1 = 0;
reg_ynm2 = 0;
for n =1:length(x)
    y(n) = x(n)*b(1) + reg_xnm1*b(2) + reg_xnm2*b(3) ...
                     - reg_ynm1*a(2) - reg_ynm2*a(3);
    reg_xnm2 = reg_xnm1;
    reg_xnm1 = x(n);
    reg_ynm2 = reg_ynm1;
    reg_ynm1 = y(n);
end
figure;
subplot(211)
plot(x); ylim([-1.2 1.2]);
subplot(212)
plot(y,'r'); ylim([-1.2 1.2]);
```

ПРИЛОЖЕНИЕ Г.
(обязательное)
Распределенная арифметика на памяти

Как известно базовой операцией цифровой обработки сигналов является вычисление выражений вида

$$y = \sum_{n=0}^{N-1} x_n \cdot A_n . \qquad (Г.1)$$

В этом выражении x_n – это входные переменные, а A_n – это фиксированные коэффициенты. Выражение (Г.1) представляет собой скалярное произведение двух *N*-мерных векторов. Оно является базовым выражением для таких фундаментальных алгоритмов ЦОС как цифровая фильтрация и Дискретное Преобразование Фурье (ДПФ). Выражения вида (Г.1) эффективно могут быть реализованы при помощи ***распределенной арифметики*** (РА).

Распределенная арифметика (англ. – Distributed Arithmetic) получила свое название в силу того, что арифметические операции представляются не в «собраном» привычном виде сумматоров и умножителей, а «распределены» в неузнаваемые формы.

Для пояснения принципа вычислений на основе распределенной арифметики рассмотрим пример вычисления (Г.1). Если входные переменные x_n представлены в дополнительном коде и нормированы так что $|x_n| < 1$, то мы можем представить каждый x_n как

$$x_n = -b_{n0} + \sum_{k=1}^{K-1} b_{nk} 2^{-k} , \qquad (Г.2)$$

где b_{nk} биты, 0 или 1, b_{n0} – знаковый бит, а $b_{n,K-1}$ – это младший значащий разряд (англ. – least significant bit (LSB)).

Теперь объединим уравнения (Г.1) и (Г.2) так, чтобы выразить y в терминах битов x_k

$$y = \sum_{n=0}^{N-1} A_n \left[-b_{n0} + \sum_{k=1}^{K-1} b_{nk} 2^{-k} \right]. \qquad (Г.3)$$

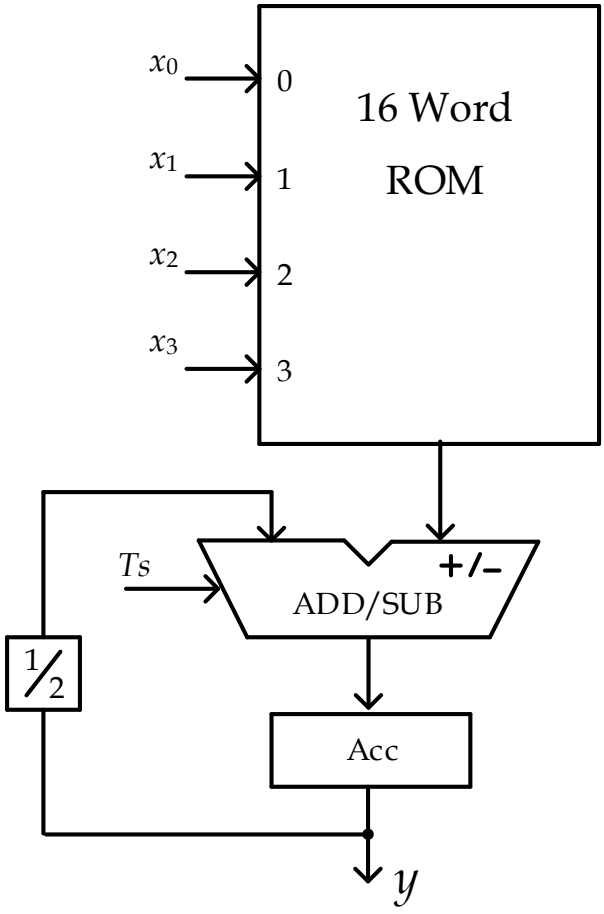
Изменив порядок суммирования мы получим

$$y = -\sum_{n=0}^{N-1} A_n b_{n0} + \sum_{k=1}^{K-1} \left[\sum_{n=0}^{N-1} A_n b_{nk} \right] 2^{-k} . \qquad (Г.4)$$

Этот шаг является ключевым: уравнение (Г.4) приводит к вычислениям на распределенной арифметике. Рассмотрим выражение в скобках из выражения (Г.4):

$$\sum_{n=0}^{N-1} A_n b_{nk} . \qquad (Г.5)$$

Из-за того, что b_{nk} принимает только значение 0 либо 1, то выражение (Г.5) может принимать только 2^N различных значений. Предпочтительнее не вычислять эти значения каждый раз, а вычислить их заранее один раз и сохранить в ПЗУ (ROM). Вычисления на распределенной арифметике, как правило, побитовые (англ. bit-serial), т. е. обрабатывается за один такт один бит входных данных. Поэтому входные данные могут быть непосредственно использованы для адресации ПЗУ (т. е. $\sum_{n=0}^{N-1} A_n b_{nk}$), данные из которого попадают в аккумулятор. После K таких циклов, аккумулятор будет содержать результат y. Знаковый разряд обрабатывается последним, поэтому как видно из формулы (Г.4), в последнем цикле содержимое памяти должно не складываться с аккумулятором, а вычитаться. Пример реализации фильтра 4-го порядка на распределенной арифметике приведён на рисунке Г.1.



b_{0k}	b_{1k}	b_{2k}	b_{3k}	ROM
0	0	0	0	0
0	0	0	1	A_3
0	0	1	0	A_2
0	0	1	1	A_2+A_3
0	1	0	0	A_1
0	1	0	1	A_1+A_3
0	1	1	0	A_1+A_2
0	1	1	1	$A_1+A_2+A_3$
1	0	0	0	A_0
1	0	0	1	A_0+A_3
1	0	1	0	A_0+A_2
1	0	1	1	$A_0+A_2+A_3$
1	1	0	0	A_0+A_1
1	1	0	1	$A_0+A_1+A_3$
1	1	1	0	$A_0+A_1+A_2$
1	1	1	1	$A_0+A_1+A_2+A_3$

Рисунок Г.1 – Реализация КИХ-фильтра 4-го порядка на РА.

Сигнал T_s на схеме обозначает сигнал приходящий одновременно со знаковым разрядом. В этот момент времени содержимое памяти вычитается из аккумулятора.

Распределенная арифметика перестает быть эффективной, если количество коэффициентов в выражении (Г.1) будет большим. Например, для 40 коэффициентов потребуется объем памяти, равный 1 Тбайту, что, разумеется, недопустимо для практической реализации. Поэтому для того чтобы снизить объем требуемой памяти предлагается следующий подход. Выражение (Г.1) разбивают на несколько подвыражений:

$$y = \sum_{n=0}^{N-1} x_n \cdot A_n = \sum_{n=0}^{N_1-1} x_n \cdot A_n + \sum_{n=N_1}^{N_2-1} x_n \cdot A_n + \ldots + \sum_{n=N_k}^{N-1} x_n \cdot A_n \,. \qquad (Г.6)$$

Затем каждую сумму произведений в правой части выражения (Г.6) реализуют при помощи распределенной арифметики, а затем результаты суммируют. Для примера, если фильтр 40 порядка разбить на 8 субфильтров 5-го порядка, то вместо 0,5 Тбайт памяти понадобиться всего лишь 8 блоков памяти по 16 байт каждый. Накладными расходами в данном случае становиться то, что приходится строить пирамиду сумматоров, для суммирования выходов всех субфильтров. Недостатком этого метода является то, что возрастает задержка распространения сигнала на пирамиде сумматоров. Пример, иллюстрирующий данный подход, приведен на рисунке Г.2, где реализуется КИХ-фильтр 6-го порядка. Импульсная характеристика фильтра разбита на два субфильтра. В данном случае пирамида сумматоров состоит всего лишь из одного сумматора.

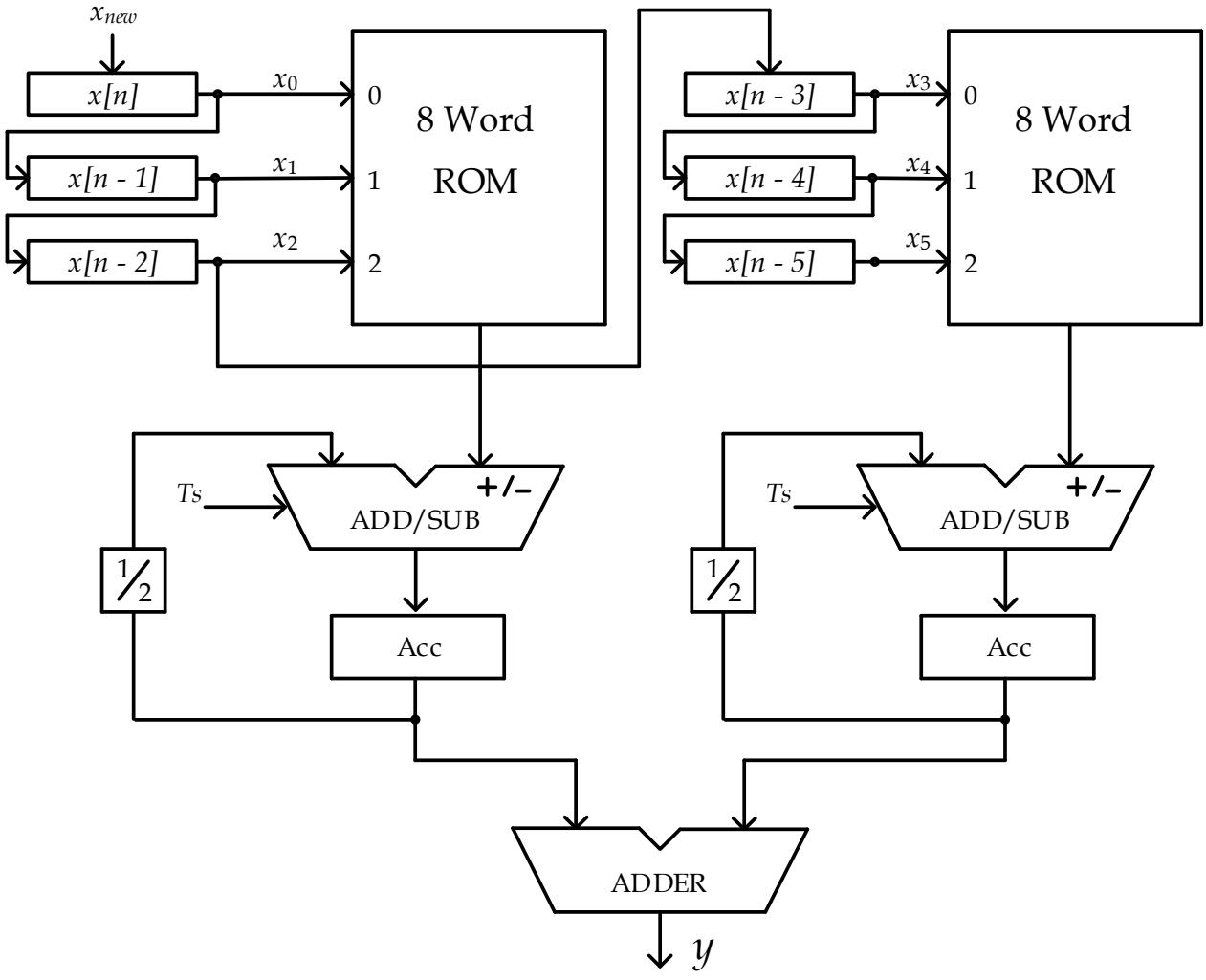


Рисунок Г.2 – Реализация КИХ-фильтра 6 порядка на РА с разбиением на субфильтры

Можно заметить, что т. к. входные данные поступают в последовательном виде (один бит за такт, или 1 BAAT – 1 Bit At A Time), то это приводит к низкой скорости вычисления выражения (Г.1). Так если входные данные это K разрядные слова, то потребуется это K тактов для формирования выходного результата.

Дополнительное увеличение скорости при вычислении на распределенной арифметике можно достичь, если разбить каждое входное слово на L подслов (L должно быть делителем K). Это приводит к тому, что возрастает размерность входного вектора в L раз. И вместо единственного блока памяти используется L блоков и расширенный аккумулятор. Этот подход иллюстрируется на рисунке Г.Г.3. В данном случае $L = 2$, и в схеме происходит обработка за один такт 2 бит, отсюда и название 2 BAAT. Получение скорости приводит к линейному увеличению объема требуемой памяти.

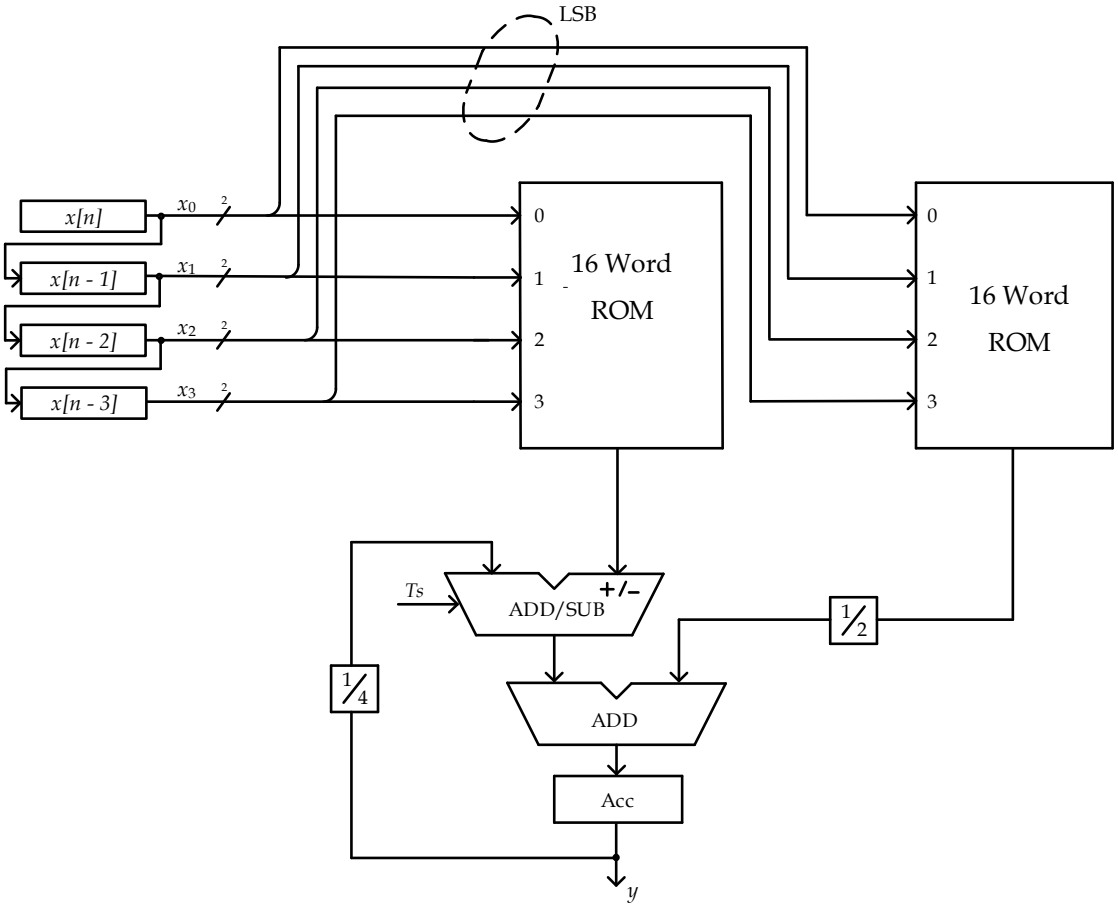


Рисунок Г.3 – Реализация КИХ-фильтра 4 порядка на РА (схема 2 BAAT)

Список использованных источников

Применение алгебры кватернионов

1 Кантор, И. Л, Гиперкомплексные числа / И. Л. Кантор, А. С. Солодовников. – М. : Наука, 1973. – С. 145.

2 Гантмахер, Ф. Р. Теория матриц 5-е изд./ Ф. Р. Гантмахер. – М. : Физматлит, 2004. – С. 560.

3 Alexiadis, D. Estimation of motions in color image sequences using hypercomplex Fourier transforms / D. Alexiadis, G. Sergiadis // IEEE Trans. Image Process. – 2009. – Vol. 18, № 1. – P. 168–187.

4 Choukroun, D. Novel quaternion kalman filter / D. Choukroun, I. Bar-Itzhack, Y. Oshman // IEEE Trans. Aerosp. Electron. Syst. – 2006. – Vol. 42, №1. – P. 174–190.

5 Miron, S. Quaternion-music for vector-sensor array processing / S. Miron, N. Le Bihan, J. Mars // IEEE Trans. Signal Process. – 2006. – Vol. 54, №4. – P. 1218-1229.

6 Sercov, V. V. Digital hypercomplex all-pass filters: A novel filters bank building block / V. V. Sercov, A. A. Petrovsky, D. V. Lushtyk // Proc. 6th Int. Workshop on Systems, Signals and Image Proc. (IWSSIP). – Bratislava, Slovakia : 1999. – P. 181–184.

7 Karney, C. Quaternions in molecular modeling / C. Karney // J. Molecular Graphics and Modelling. – 2007. – Vol. 25, №5. – P. 595–604.

8 Howell, T. D. The complexity of the quaternion product: Tech. Rep. TR 75-245 / T. D. Howell, J. C. Lafon: Cornell University, 1975. 13 P.

9 Parfieniuk, M. Quaternion multiplier inspired by the lifting implementation of plane rotations / M. Parfieniuk, A. Petrovsky // IEEE Trans. Circuits Syst. I. – 2010. – Vol. 57, №10. – P. 2708–2717.

Многоскоростная цифровая обработка сигналов

10 Vaidyanathan, P. P. Multirate Systems and Filter Banks / P. P. Vaidyanathan. – Englewood Cliffs, NJ : Prentice-Hall, 1993. – P. 911.

11 Парфенюк, М. Параунитарные банки фильтров на основе алгебры кватернионов: теория и применение / М. Парфенюк, А.А. Петровский // Цифровая обработка сигналов. – 2008. – № 1. – С. 22-36.

12 Parfieniuk, M. Inherently lossless structures for eight- and six-channel linear-phase paraunitary filter banks based on quaternion multipliers / M. Parfieniuk, A. Petrovsky // Signal Process. – 2010. – Vol. 90. – P. 1755-1767.

13 Parfieniuk, M. Quaternionic building block for paraunitary filter banks / M. Parfieniuk, A. Petrovsky // Proc. 12th European Signal Processing Conf. (EUSIPCO). – Vienna, Austria : 2004. – P. 1237-1240.

14 Suzuki, T. Generalized block-lifting factorization of M-channel biorthogonal filter banks for lossy-to-lossless image coding / T. Suzuki, M. Ikehara, T. Nguyen // IEEE Trans. Image Process. – 2012. – Vol. 21, №7. – P. 3220-3228.

15 Wavelet transforms that map integers to integers / A. R. Calderbank, I. Daubechies, W. Sweldens, B.-L. Yeo // Appl. Comput. Harmon. Anal. – 1998. – Vol. 5, №3. – P. 332–369.

16 Petrovsky, N. Pipelined embedded processor of quaternionic m-band wavelets for image multiresolution analysis / N. Petrovsky, M. Parfieniuk, A. Petrovsky // 2013 2nd Mediterranean Conference on Embedded Computing (MECO). – Budva, Monternegro : 2013. – P. 196–199.

CORDIC-алгоритмы

17 Оранский, А. М. Аппаратные методы в цифровой вычислительной технике / А. М. Оранский. – Минск : Изд. БГУ им. В.И.Ленина, 1977.

18 Volder, J. E. The CORDIC trigonometric computing technique / J. E. Volder // IRE Trans. Electron. Comput. – 1959. – Vol. 8. – P. 330–334.

19 Walther, J. S. A unified algorithm for elementary functions / J. S. Walther // Proceedings of the, Spring Joint Computer Conference. – AFIPS '71 (Spring). – New York, NY, USA: ACM, 1971. – P. 379–385.

20 50 years of CORDIC: Algorithms, architectures, and applications / P. Meher, J. Valls, T.-B. Juang [et al] // IEEE Trans. Circuits Syst. I. – 2009. – Vol. 56, № 9. – P. 1893–1907.

21 Hsiao, S. F. Parallel singular value decomposition of complex matrices using multidimensional CORDIC algorithms / S. F. Hsiao, J. M. Delosme // IEEE Trans. Signal Process. – 1996. – Vol. 44, №3. – P. 685–697.

22 Hsiao, S.-F. Householder CORDIC algorithms / S.-F. Hsiao, J.-M. Delosme // IEEE Trans. Comput. – 1995. – Vol. 44, № 8. – P. 990-1001.

23 Hsiao, S.-F. Redundant constant-factor implementation of multi-dimensional CORDIC and its application to complex SVD / S.-F. Hsiao, C.-Y. Lau, J.-M. Delosme // IEEE Trans. VLSI Syst. – 2000. – Vol. 25, № 2. – P. 155-166.

24 Meher, P. CORDIC designs for fixed angle of rotation / P. Meher, S. Park // IEEE Trans. VLSI Syst. – 2013. – Vol. 21, №2. – P. 217-228.

25 Parfieniuk M., Rapid Prototyping of Quaternion Multiplier: From Matrix Notation to FPGA-Based Circuits / M. Parfieniuk, N. Petrovsky, A. Petrovsky // Rapid Prototyping Technology / Principles and Functional Requirements, Dr. M. Hoque (Ed.), InTech, Austria, Vienna – 2011.– P. 227–246

26 Петровский, Н. А. Многомерный CORDIC алгоритм кватернионов с «разреженными» итерациями / Н. А. Петровский, М. Парфенюк // Материалы 15-й международной конференции «Цифровая обработка сигналов и её применение» (DSPA'2013). – Т. 2. – М. : 2013. –С. 206-210.

27 Петровский, Н. А. 4D-CORDIC арифметика для процессора параунитарного банка фильтров на основе алгебры кватернионов / Н. А. Петровский, М. Парфенюк, А. А. Петровский // Материалы 40-й международной научной конференции «Вопросы оптимизации вычислений»: Институт кибернетики НАН Украины. – Ялта, Украина: 2013. – С. 214-215.

28 Petrovsky, N. A. The CORDIC-inside-lifting architecture for constant-coefficient hardware quaternion multipliers / N. A. Petrovsky, M. Parfieniuk // In Proc. Int. Conf. on Signals and Electronic Systems (ICSES). – Wroclaw, Poland: 2012. – P. 6.

29 Петровский, Н. А. Рекурсивный процессор умножителя кватернионов со структурной CORDIC-лестничной параметризацией / Н. А. Петровский, А. В. Станкевич // Тезисы докладов международной научной конференции ИТС'2013 (Информационные технологии и системы) / БГУИР. – Минск, Беларусь: 2013. – С. 196–197.

30 Ключеня, В. В. Быстрое прототипирование встраиваемых программируемых систем на ПЛИС для мультимедийных приложений / В.В. Ключеня, Н.А. Петровский // Информатика. — 2015. — № 47. — С. 13–28.

31 V. Benes, Mathematical theory of connecting networks and telephone traffic, NY, Academic Press, 1965, 319 P.

Св. план 2018 , поз. 34

Учебное издание

Вашкевич Максим Иосифович
Петровский Николай Александрович
Петровский Александр Александрович

ПРОЕКТИРОВАНИЕ ВЫЧИСЛИТЕЛЬНЫХ СРЕДСТВ С ДИНАМИЧЕСКИ РЕКОНФИГУРИРУЕМОЙ АРХИТЕКТУРОЙ.

ЛАБОРАТОРНЫЙ ПРАКТИКУМ
ПОСОБИЕ

Редактор *Е. С. Юрец*
Корректор
Компьютерная правка, оригинал-макет

Подписано в печать . Формат 60x84 1/16. Бумага офсетная. Гарнитура «Таймс».
Отпечатано на ризографе. Усл. печ. л. . Уч.-изд. л. . Тираж 70 экз. Заказ 276

Издатель и полиграфическое исполнение: учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники».
Свидетельство о государственной регистрации издателя, изготовителя, распространителя печатных изданий №1/238 от 24.03.2014, №2/113 от 07.04.2014, №3/615 от 07.04.2014,
ЛП №02330/264 от 14.05.2014.
220013, Минск, П. Бровки, 6