А.В. Ворон

Реализация базовых арифметических операций в масштабированной системе счисления Бергмана (SBNS)

Аннотация. В статье рассматриваются алгоритмические особенности выполнения базовых арифметических операций в масштабированной системе счисления Бергмана (SBNS – Scaled Bergman Number System) с фиксированным весовым масштабным коэффициентом $k = 0.2$. Рассмотрены механизмы реализации линейных операций (сложения и вычитания), а также декомпозированы и детально описаны специфические аппаратные фазы выполнения нелинейных операций – умножения и деления в условиях масштабного смещения базиса. На конкретном примере численного поразрядного умножения продемонстрирована работа двухфазного арифметико-логического устройства (АЛУ) с использованием циклического доумножителя на константу 5 для масштабной коррекции кодов. Описан трехэтапный цикл деления, использующий аппаратный фильтр четности для устранения избыточности и обеспечения быстрой и однозначной поразрядной операции сравнения.

Ключевые слова: масштабированная система счисления Бергмана (SBNS); фильтр четности; правило четности; золотое сечение; нелинейные операции АЛУ.

Введение. Современная полупроводниковая микроэлектроника и вычислительная техника, базирующиеся на классической архитектуре фон Неймана и двоичном кодировании, вплотную приблизились к технологическому и физическому тупику. В качестве альтернативы в различное время были разработаны избыточные позиционные системы счисления [1, 2, 3, 4, 5, 6]. Для преодоления существующих ограничений двоичной системы счисления автором была разработана и формализована Масштабированная система счисления Бергмана (Scaled Bergman Number System, SBNS [7]), в которой осуществлен отход от привязки к абстрактной единице ($\varphi^0 = 1$) в пользу центрального элемента масштаба – весового масштабного коэффициента $k = 0.2$ [7]. Любой n-й член ряда весов SBNS задается аналитической формулой $X_n = 0.2 \times \varphi^n$ [7]. В качестве теоретического фундамента для преодоления избыточности здесь использовались принципы классической фибоначчиевой компьютерной арифметики [2, 3], а также опыт проектирования Новой фибоначчиевой системы счисления (NFNS) с дуальным базисом для помехоустойчивых сред [4, 5].

Настоящая работа посвящена исследованию базовой математической логики развертывания и свертывания разрядов на уровне арифметико-логического устройства (АЛУ) процессора SBNS [7]. Целью статьи является декомпозиция алгоритмов сложения, вычитания, поразрядного умножения с коррекцией промежуточных масштабов и трехэтапного деления, а также математическое обоснование встроенного триггерного контроля целостности данных на основе «правила четности».

Основная часть. С точки зрения математики, базовая логика развертывания и свертывания разрядов в SBNS наследуется из классической системы счисления Джорджа Бергмана [1]. Поскольку в основе обеих систем лежит иррациональное отношение золотого сечения φ, то фундаментальные тождества для переносов (такие как $\varphi^n = \varphi^{n-1} + \varphi^{n-2}$) остаются неизменными.

Тем не менее, «переписать» алгоритмы системы счисления Джорджа Бергмана для SBNS не получится. Введение фиксированного масштабного коэффициента ($k = 0.2$) создает математическое смещение, которое требует коррекции правил.

1. Линейные операции (сложение и вычитание). Здесь преобразование тривиально. Так как коэффициент 0.2 можно просто вынести за скобки, правила сложения и вычитания в системе счисления Джорджа Бергмана корректно переносятся на SBNS один к одному. В системе счисления Джорджа Бергмана «сворачивание» имеет вид $1\varphi^n + 1\varphi^{n-1} \rightarrow 1\varphi^{n+1}$, а в SBNS – $1 \times (0.2\varphi^n) + 1 \times (0.2\varphi^{n-1}) \rightarrow 1 \times (0.2\varphi^{n+1})$. Логика переносов и заемов в АЛУ не меняется, меняется только физический «вес», закрепленный за каждым триггером регистра.

2. Нелинейные операции (умножение). В классической системе Джорджа Бергмана умножение двух базовых разрядов имеет вид $\varphi^a \times \varphi^b = \varphi^{a+b}$. Но в SBNS при перемножении весов возникает квадрат масштабного коэффициента $X_a \times X_b = (0.2 \times \varphi^a) \times (0.2 \times \varphi^b) = 0.04 \times \varphi^{a+b}$. Результат как бы «вылетает» из базовой шкалы системы, так как нам нужно получить значение в форме $0.2 \times \varphi^n$. Поэтому при представлении умножения нужно описать алгоритм масштабной коррекции. Для этого процессору нужно выполнить сдвиг полученного кода частичного произведения. Потребуется ввести дополнительную операцию деления промежуточного результата на 0.2 (что эквивалентно умножению на целое число 5) для восстановления исходного базиса SBNS.

3. Деление. Классическое деление в системе счисления Джорджа Бергмана опирается на пошаговое вычитание делителя, сдвинутого на определенное число разрядов φ^{-i}. В SBNS этот сдвиг также смещается на величину масштаба. Каждый шаг деления или вычисления тригонометрии (через CORDIC) теперь должен оперировать константами, масштабированными под шаг $0.2 \times \varphi^{-i}$. Арифметику системы счисления Джорджа Бергмана мы дополняем масштабным АЛУ-контроллером, который компенсирует смещение (k=0.2) при умножении, взамен получая аппаратное самозатухание дробных шумов.

Методика поразрядного умножения в SBNS (на примере 2 × 2 = 4). Поразрядное умножение в системе SBNS на конкретном примере, и как классический алгоритм Джорджа Бергмана модифицируется из-за масштабного коэффициента (k = 0.2). Для простоты и наглядности возьмем умножение двух целых чисел: 2 × 2 = 4.

Шаг 1. Перевод чисел в коды SBNS. Число 2 в системе SBNS представляется тремя активными разрядами (для Фильтрующего Рабочего Кода «А») (2 = 1.37082039325 + 0.52360679775 + 0.12360679775). В терминах весов это имеет вид $2 = X_4 + X_2 + X_{-1}$. Запишем наше выражение в виде весов SBNS $(X_4 + X_2 + X_{-1}) \times (X_4 + X_2 + X_{-1})$.

Шаг 2. Поразрядное перемножение (формирование матрицы). Раскрываем скобки, перемножая каждый разряд с каждым, как в обычном полиноме. В классической системе Джорджа Бергмана мы бы просто сложили индексы степеней ($\varphi^a \times \varphi^b = \varphi^{a+b}$). Но в SBNS каждый вес содержит 0.2. $X_a \times X_b = (0.2\varphi^a) \times (0.2\varphi^b) = 0.04\varphi^{a+b}$. Составим матрицу частичных произведений (индексы просто складываются, но перед каждым результатом в уме держим коэффициент 0.2).

$$X_4 \times X_4 = 0.2 \times X_8$$
$$X_4 \times X_2 = 0.2 \times X_6$$
$$X_4 \times X_{-1} = 0.2 \times X_3$$
$$X_2 \times X_4 = 0.2 \times X_6$$
$$X_2 \times X_2 = 0.2 \times X_4$$
$$X_2 \times X_{-1} = 0.2 \times X_1$$
$$X_{-1} \times X_4 = 0.2 \times X_3$$
$$X_{-1} \times X_2 = 0.2 \times X_1$$
$$X_{-1} \times X_{-1} = 0.2 \times X_{-2}$$

Шаг 3. Предварительное суммирование. Складываем одинаковые разряды (например, две штуки $0.2 \times X_6$ дают $0.4 \times X_6$). Промежуточный результат = $0.2 \times (X_8 + 2X_6 + 2X_3 + X_4 + 2X_1 + X_{-2})$. Теперь, используя базовое фибоначчиево правило свертки системы счисления Джорджа Бергмана ($X_n = X_{n-1} + X_{n-2}$), АЛУ процессора упрощает эту комбинацию весов. Например, два X_6 можно развернуть, а затем свернуть с другими разрядами. После выполнения стандартных бергмановских сверток/разверток комбинация в скобках преобразуется к красивому виду $(X_8 + 2X_6 + 2X_3 + X_4 + 2X_1 + X_{-2}) \rightarrow X_9 + X_7 + X_4 + X_2 + X_{-1} + X_{-3}$. Подставим реальные числовые значения весов из препринта, чтобы проверить точность. 15.2026311235 (X_9) + 5.80688837075 (X_7) + 1.37082039325 (X_4) + 0.52360679775 (X_2) + 0.12360679775 (X_{-1}) + 0.0472135955 (X_{-3}) = 23.0747670785. Умножаем на вынесенный коэффициент 0.2. 23.0747670785 × 0.2 = 4.6149534157. Это промежуточное значение в промежуточном масштабе k=0.04 (в пересчете на базовый базис). Для восстановления истинного кода числа 4 требуется фаза масштабной коррекции (Шаг 4).

Шаг 4. Масштабная коррекция (умножение на 5). В этот момент классический алгоритм Джорджа Бергмана завершился бы, но в SBNS АЛУ обязано вернуть число в исходную шкалу. Для этого промежуточный результат нужно разделить на 0.2, что физически означает умножить на целое число 5. Как аппаратно умножить код на 5 в фибоначчиевой логике? Число 5 – это вес разряда $X_5/0.2$ или сумма $X_4 + X_2 + X_{-1} + 1$. Процессор выполняет быстрое аппаратное умножение промежуточного кода на константу 5. $4.6149534157 \times 5 = 23.0747670785$.

Шаг 5. Финальная фильтрация и «Затухание шума». Полученное значение 23.0747670785 раскладывается аппаратным фильтром обратно в сетку весов SBNS. 23.0747670785 точно соответствует сумме весов $X_9 + X_7 + X_4 + X_2 + X_{-1} + X_{-3}$. Посмотрим в таблицу кодов автора (Table A3) для числа 4. Для числа 4 Фильтрующий Рабочий Код «A» (FWC) имеет единицы в разрядах X_7, X_1, X_{-2}, X_{-4}, X_{-6}, X_{-8}. При переходе к Контрольному коду «B» (FCC) или выполнении финишной свертки, дробные «хвосты» чисел (все эти концевые пятерки и шумы усечения регистров) согласно Теореме о симметрии усечения взаимно гасят друг друга.

Умножение в SBNS «работает» по законам системы счисления Джорджа Бергмана, но требует двухфазного АЛУ.

1. Фаза № 1. Сложение индексов разрядов (система счисления Джорджа Бергмана) с автоматическим сдвигом результата в «промежуточную» шкалу k=0.04.

2. Фаза № 2. Аппаратный циклический доумножитель на фиксированную константу 5 (масштабный корректор) перед тем, как отправить код в аппаратный фильтр четности.

Это усложняет АЛУ на несколько логических вентилей по сравнению с вариантом в классической системе счисления Джорджа Бергмана, но взамен полностью убирает проблему бесконечных дробей при округлениях.

Операция деления в системе SBNS – это наиболее сложный, но математически предсказуемый процесс. Поскольку базовые веса разрядов жестко привязаны к геометрии золотого сечения φ, деление в SBNS наследует классические принципы деления Джорджа Бергмана, но модифицируется из-за масштабного коэффициента $k = 0.2$.

В традиционных процессорах деление выполняется либо как последовательное вычитание делителя из делимого со сдвигом регистра (алгоритмы семейства *Restoring/Non-Restoring Division*), либо через умножение на обратную величину (метод Ньютона-Рафсона) [8]. В SBNS процесс делится на три аппаратных этапа.

Этап 1. Выравнивание масштабов (предобработка). Так как каждый базовый разряд в SBNS имеет встроенный масштаб $X_n = 0.2 \times \varphi^n$, при делении одного числа на другое в аналитическом виде возникает математическое смещение $X_a / X_b = 0.2 \times \varphi^a / 0.2 \times \varphi^b = \varphi^{a-b}$. При этом масштабный коэффициент 0.2 в числителе и знаменателе взаимно сокращается. На выходе мы получаем «чистую» степень золотого сечения φ^{a-b}, которая вылетает из стандартной сетки весов SBNS (где каждый разряд обязан содержать множитель 0.2).

Прежде чем запустить цикл деления, АЛУ процессора должно скорректировать делимое. Оно аппаратно умножает делимое на 0.2 (или делит на 5), чтобы промежуточный результат деления сразу формировался в правильном базисе SBNS $(0.2 \times X_a) / X_b = 0.2 \times (0.2 \times \varphi^a) / 0.2 \times \varphi^b = 0.2 \times \varphi^{a-b} = X_{a-b}$.

Этап 2. Цикл вычитания и сдвига (аналог алгоритма системы счисления Джорджа Бергмана). После выравнивания масштабов запускается классическая процедура, напоминающая деление «уголком», которая движется от старших разрядов к младшим:

1. Определение знака и сравнение. Процессор сопоставляет старшие разряды делимого (остатка) и делителя.

2. Вычитание. Если остаток больше делителя, в соответствующий разряд частного записывается единица (1), а из остатка вычитается делитель.

3. Сдвиг. Если остаток меньше, в разряд частного пишется ноль (0), а делитель сдвигается на один разряд вправо (что эквивалентно его делению на φ). В SBNS сдвиг кода влево или вправо на один триггер означает автоматическое умножение или деление веса на φ, так как связь между соседними ячейками строго геометрическая: $X_n \times \varphi = X_{n+1}$.

Главная аппаратная сложность этого этапа заключается в том, что в двоичной системе определить, какое число больше, очень легко – достаточно сравнить первые несовпадающие биты слева направо. В избыточных системах [1, 2] (Джорджа Бергмана, Фибоначчи, NFNS [4, 5], SBNS) одно и то же число может быть записано множеством способов [6]. Например, число может быть записано как старший разряд 100... или как сумма младших 011.... В этом отношении в SBNS (а также – в NFNS [4, 5]) существует значительное преимущество – аппаратный фильтр четности [7]. Поскольку фильтр строго ограничивает избыточность и удерживает код только в двух детерминированных состояниях (Рабочем «A» или Контрольном «B»), то процессору не нужно тратить такты на промежуточные перекодирования. Фильтр делает операцию сравнения кодов быстрой и однозначной.

Этап 3. Автоматическая остановка цикла и гашение остатка. В двоичных процессорах деление дробных чисел часто превращается в бесконечную периодическую дробь (например, 1 / 3 = 0.3333...), и цикл останавливается принудительно, когда заканчивается разрядность регистра, что порождает погрешность.

В SBNS при делении целых чисел или соизмеримых дробей в диапазоне, благодаря свойству «взаимного уничтожения концевых пятерок», остаток деления на определенном шаге (на 10–12 десятичном знаке) генерирует микроструктуру, которая при очередной фибоначчиевой свертке обращается в точный ноль. Процессор физически «видит», что вычислять дальше нечего (остаток равен 0), и мгновенно прекращает цикл деления. В этот же момент срабатывает алгоритм динамической разрядности, отсекая пустые ячейки регистра для экономии энергии.

Операция аппаратного контроля целостности в системе SBNS – это специализированная логическая процедура, которая выполняется непосредственно на уровне триггеров арифметико-логического устройства (АЛУ). Она служит для мгновенного обнаружения аппаратных сбоев (например, вызванных космической радиацией в аэрокосмических системах) без привлечения программных избыточных кодов (таких как CRC или контрольные суммы) [9]. Эта операция базируется на сформулированной автором теореме об инвариантности четности в условиях работы аппаратного фильтра [7]. Логика контроля опирается на уникальное свойство SBNS (NFNS [4, 5]): любое число внутри процессора всегда «зажато» аппаратным фильтром между двумя валидными кодовыми состояниями.

1. Рабочий код «A» (Развернутая форма представления числа).

2. Контрольный код «B» (Свернутая форма представления числа).

Позиции любых двух соседних активных единиц в «правильном» коде обязаны иметь противоположную четность, если они стоят рядом или разделены исключительно *четным* числом нулей. Сумма усеченных весов активных разрядов для кодов «A» и «B» строго инвариантна (равна единому целому значению). При переходе из формы A в форму B и обратно погрешность транзита равна абсолютному нулю (EpsilonTransition≡0).

Заключение.

1. В то время как линейные операции (сложение, вычитание) переносятся из классического алгоритма системы счисления Джорджа Бергмана на SBNS один к одному за счет вынесения коэффициента 0.2 за скобки. В то же время нелинейные операции требуют введения масштабного АЛУ-контроллера. Умножение сопровождается возникновением квадрата масштаба $0.04 \times \varphi^{a+b}$, что устраняется введением второй фазы вычислений – быстрого аппаратного доумножения кода частичного произведения на фиксированную константу 5. Это делается для восстановления исходного базиса системы.

2. Традиционная проблема медленного поразрядного сравнения остатка и делителя, свойственная избыточным иррациональным системам, в SBNS полностью решена применением аппаратного фильтра (правила четности). Фильтр жестко ограничивает избыточность кодирования, удерживая число строго в рамках двух детерминированных кодовых состояний – Рабочего Кода «A» и Контрольного Кода «B», что делает операцию сравнения битов при пошаговом вычитании однозначной и быстродействующей.

3. Опираясь на теорему об инвариантности четности, АЛУ способно выполнять мгновенный сквозной контроль целостности данных непосредственно на уровне триггеров регистра без привлечения программной избыточности (CRC). Погрешность транзита при переходе из формы «А» в форму «В» равна абсолютному нулю, что обеспечивает высокую помехоустойчивость вычислительной среды при воздействии внешних отрицательных факторов.

Список литературы:

1. Бергман Дж., Система счисления с иррациональным основанием (A Number System with an Irrational Base) // Mathematics Magazine. 1957. Vol. 31, № 2. P. 98–110.

2. Стахов А.П., Введение в алгоритмическую теорию измерения. М.: Советское радио, 1977. 288 с.

3. Стахов А.П., Коды золотой пропорции. М.: Радио и связь, 1984. 152 с.

4. Ворон А.В., Избыточность кода Фибоначчи и разработка новой системы счисления Фибоначчи – NFNS // «Академия Тринитаризма», М., Эл № 77-6567, публ. 28623, 25.07.2026.

5. Ворон А.В., Разработка избыточной системы счисления Фибоначчи NFNS с дуальным базисом для помехоустойчивых вычислительных сред // «Академия Тринитаризма», М., Эл № 77-6567, публ. 30171, 21.09.2026.

6. Цекендорф Э., Представление натуральных чисел в виде суммы чисел Фибоначчи или чисел Люка (Représentation des nombres naturels par une somme de nombres de Fibonacci ou de nombres de Lucas) // Bulletin de la Société Royale des Sciences de Liège. 1972. Vol. 41. P. 179–182.

7. Ворон А.В., Масштабированная система счисления Бергмана (SBNS) и фильтр Ворона как новый базис вычислительной техники // «Академия Тринитаризма», М., Эл № 77-6567, публ. 30204, 22.10.2026.

8. Корен И., Алгоритмы компьютерной арифметики (Koren I. Computer Arithmetic Algorithms). 3-е изд. Бока-Ратон: AK Peters/CRC Press, 2018. 312 с.

9. Пархами Б., Компьютерная арифметика: сложность и проектирование отказоустойчивых систем (Parhami B. Computer Arithmetic: Complexity and Design Fault-Tolerant Systems). Нью-Йорк: Oxford University Press, 2010. 496 с.