**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
ДОНЕЦКОЙ НАРОДНОЙ РЕСПУБЛИКИ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

На правах рукописи

УДК 004.925.83

Иваница Сергей Васильевич

ОБОСНОВАНИЕ ЗАКОНОМЕРНОСТЕЙ, АРИФМЕТИКО-ЛОГИЧЕСКИХ АЛГОРИТМОВ И СТРУКТУР СИСТЕМ КОМПЬЮТЕРНОЙ ОБРАБОТКИ ИНФОРМАЦИИ

Специальность 05.13.01 – Системный анализ, управление и обработка информации (по отраслям) (технические науки)

Диссертация на соискание ученой степени
кандидата технических наук

Научный руководитель:
кандидат технических наук, доцент,
Аноприенко Александр Яковлевич

Идентичность всех экземпляров
ПОДТВЕРЖДАЮ
Ученый секретарь
диссертационного совета Д 01.024.04
канд.тех.наук Т.В. Завадская

Донецк – 2019

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы исследований. Постоянное увеличение насыщенности современной техносферы компьютерными технологиями сопряжено с практически экспоненциальным ростом объемов вычислений. Это существенно актуализирует вопросы обеспечения эффективности и надежности средств и методов современного компьютинга, направленного на повышение точности вычислений в процессе моделирования и проектирования сложных динамических систем.

Установлено, в частности, что многие техногенные катастрофы последних десятилетий были в первую очередь обусловлены разного рода недостатками компьютерных систем управления и вычислительными ошибками. Но в большинстве случаев ошибки в вычислениях просто остаются незамеченными, существенно искажая полученные результаты.

В связи с этим следует признать, что в настоящее время созрели все предпосылки для существенной модификации всей системы компьютерных вычислений с целью повышения ее надежности и соответствию требованиям, предъявляемым к современному компьютингу. При этом речь может идти о дальнейшем развитии как логической, так и вычислительной составляющей современного компьютинга. Особо актуальной является разработка такой модификации вычислений с плавающей запятой, которая позволила бы исключить потерю точности (и информации о ней) при представлении входных, промежуточных и результирующих значений в форматах с плавающей запятой.

В связи с этим обоснование закономерностей и перспектив арифметико-логических основ систем компьютерной обработки информации является актуальной научно-технической задачей.

Степень разработанности темы исследования. Теоретической базой для проведения исследований стали работы ведущих отечественных и зарубежных учёных и их учеников:

– в области системного анализа и системного подхода: Ракитов А. И. [1], Чернышов В. Н. [2], Вентцель Е. С. [3, 4], Антонов А. В. [5, 6], Волкова А. С. [7, 8], Садовский В. Н. [9], О'Коннор, Дж. [10], Блауберг И. В. [11, 12], Громов Ю. Ю. [13] и др.;

– в области построения логических моделей и расширения кодо-логического базиса: Заде Л. А. [14, 15, 16], Лукасевич Я. [17], Бочвар Д. А. [18], Клини С. К. [19, 20], Белнап Н. Д. [21, 22, 23], Горбатов А. В. [24], Зверев Г. Н. [25, 26, 27], Аноприенко А. Я. [28, 29, 30, 31, 32], и др.

– в области математического и интервального анализа, интервальной арифметики: Калмыков С. Л. [33], Алефельд Г. [34], Хансен Е. [35], Мур Р. [36, 37], Вощинин, А. П. [38], Шокин Ю. И. [39], Назаренко Т. И. [40], Шарый С. П. [41] и др.

– в области компьютерной обработки информации, достоверности и надежности компьютерных расчетов: Брадис В. М. [42], Никель К. [43], Яблонский С. В. [44], Канторович Л. В. [45], Юровицкий В. М. [46, 47], Петров Ю. П. [48, 49], Литвинов Г. Л. [50] и др.

Несмотря на значительный объем исследований в каждой из указанных научных областей, актуальность темы диссертации остается значимой, так как в используемой литературе не прослеживаются связи, например, между k-значными логиками и вопросами достоверности и надежности компьютерных вычислений, а, тем более, с интервальным подходом представления цифровой информации. Кроме того, лишь единицы научных работ рассматривают исследуемую сложную систему (в частности, вычислительную) в комплексе, отвечая на вопросы достоверности вычислений, и решая при этом всю последовательность задач: начиная от вопросов представления (кодирования) информации, ее обработки (арифметика и логика), до проблем, связанных с получением достоверных результатов. В качестве типичного примера такого рода можно привести полином Румпа (впервые опубликован в 1988 году), который при определенном сочетании значений переменных дает заведомо неправильный результат при всех

стандартных значениях точности вычислений с плавающей запятой практически на всех современных компьютерных системах.

Целью исследования является совершенствование современных компьютерных технологий путем расширения кодо-логического базиса с последующей реализацией новых подходов к модернизации систем компьютерной обработки информации.

Для достижения поставленной цели сформулированы и решены следующие **задачи**:

1. Аналитический обзор исследований в области арифметико-логических основ компьютерной вычислительной техники при выполнении компьютерных вычислений с выявлением достоверности полученных результатов.

2. Разработка расширенного кодо-логического базиса как следующего этапа развития арифметико-логических систем.

3. Разработка специализированных форматов чисел с плавающей запятой, использующих постбинарное кодирование для хранения и обработки числовой информации с возможностью получения достоверных результатов.

4. Разработка новых инструментов компьютерных вычислений с применением постбинарной логики и постбинарного кодирования.

5. Модификация существующих алгоритмов для реализации арифметических операций над числами при использовании тетракодирования в специализированных форматах с плавающей запятой.

Объект исследования – аппаратно-программные средства компьютерных технологий, предназначенные для решения задач обработки информации (вычислений).

Предмет исследования – кодо-логический базис, постбинарная логика и кодирование информации для выполнения арифметических и логических преобразований.

Методология и методы исследований. Исследования, проведенные в работе, базируются на системном анализе, теории информации, формальных логических моделях представления и обработки информации, аксиоматическом

аппарате теории множеств, оптимизации и обработки информации, теории кодирования, принципах построения логических систем, вещественной арифметики.

Теоретической базой исследования арифметико-логических основ компьютерной обработки информации явились научные труды отечественных и зарубежных авторов, посвященные проблемам кодо-логического базиса, вычислений чисел в форматах с плавающей запятой, используемых в современных компьютерных технологиях.

При проведении исследований, для получения анализируемых результатов проводились вычисления с помощью ЭВМ. Применялись как системы компьютерной алгебры, так и разработанное в рамках исследования прикладное программное обеспечение.

Научная новизна полученных результатов заключается в следующем:

1. В качестве компьютерного логического базиса предложена тетралогика как логика четырех состояний, которая кроме классических состояний «истина» и «ложь» использует новые логические состояния «неопределенность» и «множественность». Для тетралогики предложены и обоснованы классы логических n-арных функций (тетрафункций).

2. При переходе от тетралогики к тетракодам определены принципы кодирования/декодирования исходных числовых величин. Показано, что принципы кодирования тетракодов имеют интервальную природу, т. е. способны хранить интервальные значения в поле одного тетракода-операнда.

3. Впервые разработаны базовые арифметические операции над тетракодами (тетраарифметика) с приведением доказательств основных алгебраических тождеств.

4. Предложены модифицированные (постбинарные) форматы чисел с плавающей запятой, использующие бинарные коды и тетракоды, а также хранящие в поле формата информацию о типе чисел (число, обыкновенная дробь, интервал) и способе их кодирования (бинарное, постбинарное). Показаны оценки максимальной погрешности при кодировании чисел в модифицированные

форматы, которые не уступают значениям погрешностей, возникающих при кодировании и обработки чисел в классических форматах чисел с плавающей запятой.

5. Впервые предложен новый способ округления чисел в постбинарных форматах чисел с плавающей запятой, при котором в одном поле формата записываются значения двух точек разрядной сетки, между которыми находится исходное действительное число. Такой подход позволяет либо избежать погрешностей, появляющихся в результате округления чисел, либо снизить погрешности при кодировании в два раза (зависит от положения реального числа по отношению к разрядной сетке используемого формата).

Теоретическая и практическая значимость работы. Теоретическая значимость результатов работы заключается в определении операций и функций новых логики и арифметики, на основе которых возможны разработки аппаратурных компонентов, основанных на классических логических примитивах, что в дальнейшем может быть использовано в качестве аппаратной основы компьютерных систем следующих поколений.

Практическое значение результатов исследований:

1. Разработан необходимый и достаточный логический базис для дальнейшей реализации операционных аппаратных компонентов (управляющие схемы, сумматоры, арифметико-логические устройства, устройства памяти и пр.).

2. Определены основные арифметические операции над тетракодами, при применении которых появляется возможность создания алгебраических специализированных модулей для расчета практических вычислительных задач любой сложности.

3. Разработаны алгоритмы постбинарного округления чисел, уточнены алгоритмы арифметики с плавающей запятой, которые применяются к постбинарным типам данных в виде числовых, дробных и интервальных значений.

4. Приведены рекомендации к использованию модифицированных форматов чисел с плавающей запятой в виде спецификаций, в которых указаны размерности каждого поля и рассчитаны необходимые числовые диапазоны.

Практическое значение полученных результатов подтверждается внедрением в государственном предприятии «СТИРОЛ» (справка о внедрении № 327/4 от 18.10.19 г. выдана ГП «СТИРОЛ», г. Горловка), а также в учебный процесс ГОУВПО «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» (справка № 01-508/27 от 16.10.19 г. принята к внедрению в учебный процесс при чтении лекций и проведении практических занятий по дисциплине «Арифметико-логические основы цифровых автоматов» для студентов направления подготовки 09.03.01 «Информатика и вычислительная техника»).

Научные положения, выносимые на защиту:

1. Обоснование тетралогики как логики четырех состояний с получением базовых функций (тетрафункций), которая позволит значительно расширить возможности формализованной логической оценки разнообразных реальных процессов и результатов, что является существенным шагом к «очеловечиванию» машинной логики.

2. Новые принципы кодирования информации – тетракодирование с методами приведения новых кодов (тетракодов) к интервальным значениям; определение и свойства арифметических операций над тетракодами, что позволяет использовать тетракодирование для обеспечения достоверных вычислений путем уменьшения погрешности округления при компьютерной обработке информации.

3. Обоснованы варианты реализации постбинарных форматов с плавающей запятой и способы представления вещественных чисел в постбинарных форматах, использование которых позволяет повысить точность представления числовых значений на всех этапах компьютерной обработки информации.

Полученные результаты, положения и выводы отвечают соответствующим требованиям паспорта специальности 05.13.01 – Системный анализ, управление и обработка информации (по отраслям) (технические науки), в частности: п. 4 – «Разработка методов и алгоритмов решения задач системного анализа, оптимизации, управления, принятия решений и обработки информации»; п. 5 – «Разработка специального математического и алгоритмического обеспечения систем анализа, оптимизации, управления, принятия решений и обработки информации»; п. 8 – «Теоретико-множественный и теоретико-информационный анализ сложных систем».

Публикации. По теме диссертации опубликовано 23 научных труда в научно-технических журналах, сборниках докладов, научно-технических сборниках, научных изданиях, в том числе: 2 монографии; 2 статьи в научных изданиях, включенных в Перечень ВАК Донецкой Народной Республики; 9 статей в научных изданиях, включенных в Перечень ВАК Украины; 6 публикаций по материалам научных конференций.

Апробация результатов диссертации. Основные положения и научные результаты диссертационной работы докладывались и обсуждались на научных конференциях: IV Международная научно-техническая конференция студентов, аспирантов и молодых ученых «Моделирование и компьютерная графика» (г. Донецк, 2011 г.); II Всеукраинская научно-техническая конференция студентов, аспирантов и молодых ученых «Информационные управляющие системы и компьютерный мониторинг» (г. Донецк, 2011 г.); VII Международная научно-техническая конференция студентов, аспирантов и молодых ученых «Информатика и компьютерные технологии» (г. Донецк, 2011 г.); Международная научно-техническая конференция «Искусственный интеллект. Интеллектуальные системы» (пос. Кацивели, АР Крым, 2012 г.); V Международная научно-техническая конференция студентов, аспирантов и молодых ученых «Информационные управляющие системы и компьютерный мониторинг» (г. Донецк, 2014 г.); I Международная научно-практическая конференция

«Инновационные перспективы Донбасса: Компьютерные науки и технологии» (г. Донецк, 2015 г.).

Структура и объем работы. Диссертационная работа содержит 226 страниц машинописного текста и состоит из введения, пяти глав, заключения, списка литературы из 155 источников и 10 приложений. Текст диссертации иллюстрируется 54 рисунками и содержит 23 таблицы.

ГЛАВА 1
СОСТОЯНИЕ ВОПРОСА РАЗВИТИЯ АРИФМЕТИКО-ЛОГИЧЕСКИХ ОСНОВ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

1.1 Обзор и анализ основных исследований и разработок в области обеспечения контроля точности и достоверности результатов современного компьютинга

Многие техногенные катастрофы последних десятилетий, причину которых традиционно объясняли преимущественно «человеческими факторами», обусловлены разного рода недостатками компьютерных систем управления и вычислительными ошибками. Этот факт подтвержден при анализе целого ряда произошедших в последнее время аварий и катастроф. При этом выявлено, что вычислительные методы, используемые при проектировании и реализованные с помощью современных и передовых программных систем, обладают существенными недостатками. Главный из этих недостатков – ошибочные результаты компьютерных вычислений без извещения пользователя об их недостоверности [48].

В компьютерных вычислениях, в процессе реализации и получении результатов можно выделить три основных источника ошибок [51]:

1) погрешность исходных данных; ошибки, связанные с вводом неточных (или неверных) исходных данных;
2) ошибки, связанные с неверным выбором численных методов и алгоритмов;
3) ошибки, связанные с кодированием и обработкой числовой информации при ограниченности разрядной сетки компьютера.

Устранение или уменьшение этих ошибок в различных их сочетаниях приводит к повышению эффективности компьютерных вычислений.

На сегодняшний день аппаратное обеспечение компьютеров ориентировано на операции с числами, представленными всего в двух форматах: целые числа и

числа с плавающей запятой. Если аппаратная часть исправна, а программы не содержат ошибок, то целочисленная арифметика и программные надстройки над ней работают без погрешностей. Например, на основе целочисленной арифметики можно построить арифметику систематических дробей произвольной разрядности и обыкновенных дробей. Для точного представления результатов целочисленных арифметических операций современный компьютер имеет возможность сохранять необходимое количество разрядов для точного представления достаточно больших целых чисел. Но при решении большинства практических задач главным недостатком целочисленной арифметики является крайняя ограниченность диапазона представления чисел, что определило переход на использование в современных компьютерных технологиях преимущественно вещественных чисел.

Произвольное вещественное число представляется бесконечной десятичной или двоичной дробью. На практике в научных и инженерных вычислениях вещественные числа приходится представлять в компьютере конечными дробями, чаще всего числами в формате с плавающей запятой. Общепринятым средством для выполнения научных и инженерных вычислений на компьютерах является арифметика чисел с плавающей запятой, которая поддерживается аппаратным обеспечением компьютеров. Данный формат разработан ассоциацией IEEE и используется для представления действительных чисел в двоичном коде [52].

Каждая отдельная операция с плавающей запятой не всегда обеспечивает абсолютную точность результата, а максимальная точность операции обеспечивается в том смысле, что получаемый округленный результат может отличаться от точного вещественного значения не более чем на единицу последнего разряда мантиссы [53]. Неизбежность округления чисел с плавающей запятой порождена представлением бесконечного множества вещественных чисел конечной разрядной сеткой компьютера. Таким образом, результат множественных последовательных операций может не содержать уже ни одной верной цифры. С учетом достижения современными компьютерами способности выполнять порядка 10^{12} операций с плавающей запятой в секунду, достоверность

вычисляемых результатов, которая соответствует правильным результатам математического моделирования, становится предметом повышенного внимания.

1.1.1 Анализ особенностей вычислений, приводящих к получению недостоверных результатов в компьютерных вычислениях

Одним из простейших путей минимизации ошибок, связанных с округлением и потерей точности ввиду ограничений, обусловленных особенностью представления чисел в современных цифровых компьютерах, является дальнейший рост разрядности вычислений. В августе 2008 года был опубликован стандарт IEEE 754-2008 [54], который заменил ранее действовавший стандарт вычислений с плавающей запятой IEEE 754-1985. Стандарт 1985 года предусматривал 2 типа чисел с плавающей запятой: одинарной (32-разрядные) и двойной (64-разрядные) точности. Одной из первых его аппаратных реализаций стал арифметический сопроцессор Intel 8087, в котором дополнительно в качестве внутреннего формата использовался расширенный 80-разрядный формат с 64-разрядной мантиссой и 16-разрядным порядком. Основным нововведением в стандарте 2008 года стал формат четверной точности (128 разрядов), используемый для обеспечения повышенной точности вычислений.

Однако увеличение разрядности не в состоянии устранить все погрешности вычислений с плавающей запятой. Например, впервые опубликованный в 1988 году полином Румпа [55] при определенном сочетании значений аргументов (a = 77617, b = 33096) возвращает заведомо неправильный результат:

$$f = 333{,}75\, b^6 + a^2\,(11a^2\, b^2 - b^6 - 121b^4 - 2) + 5{,}5b^8 + a/(2b), \qquad (1.1)$$

Для различной разрядности вычислений с плавающей запятой на вычислительной системе IBM 370 были получены одинаковые, отличающиеся лишь точностью представления, результаты:

32-bit: f = 1,172604;

64-bit: f = 1,1726039400531786;

128-bit: f = 1,172603940053178631858834904520183 8.

Во всех случаях эти результаты существенно отличались от правильного, полученного с помощью специального алгоритма:

$$f = -0{,}827396059946821368141165095479816\ldots$$

В 2001 году опубликованы результаты исследования примера Румпа с применением различных современных вычислительных платформ и инструментов [56]. Вычисление полинома Румпа на компьютерах с процессорами Intel Pentium и Sun Sparc дало различные результаты в зависимости от того, какая система компьютерной алгебры (СКА) использовалась. В 2011 году в работе [57] продемонстрирована ошибочность решения полинома Румпа современными СКА и выполнен детальный анализ полинома. При этом проанализирована возможность использования интервальных методов для минимизации отрицательных последствий получения ошибочных результатов вычислений. В частности, показано, что при расчете результат компьютерных вычислений части полинома (1.1), а именно

$$333{,}75\, b^6 + a^2\,(11a^2\, b^2 - b^6 - 121b^4 - 2) + 5{,}5b^8$$

равен нулю, хотя реально имеет ненулевое значение (на рисунке 1.1 значение переменной *fl*). Это связано с особенностью вещественной арифметики стандарта IEEE 754-2008: в результате выравнивания порядков теряется значимость мантиссы одного из операндов. Поэтому возвращается неправильный результат, который фактически равен $a/(2b) = 1{,}172604\ldots$

Еще одним примером того, как на результат влияет округление и погрешность при выполнении простейших арифметических операций, является рекуррентное соотношение Мюллера. Его приводит канадский ученый в области вычислительной математики Уильям Мортон Кэхэн в своей работе [58]:

$$f(x, y) = 108 - \frac{815 - 1500 / y}{x},$$

при $x_0 = 4.00,\ x_1 = 4.25,\ x_i = f(x_{i-1}, x_{i-2})$. Параметры соотношения подобраны таким образом, что: $\lim\limits_{x \to +\infty} x_i = 5.00$. Однако, начиная с x_{14} решение становится неверным.

Таким образом, например, для $i = 80$ решение стандартными средствами C++ возвращает следующие значения:

– для формата float: $x_{80} = 100$;

– для формата double:

$$x_{80} = -99.518156613437184887516195885837078094482421875;$$

– для формата long double:

$$x_{80} = -99.518156613437176685743601467493135714903473854064941406 25.$$

Этот результат не соответствует истинному решению задачи:

$$x_{80} = 4.9999999999999999964263\ldots$$

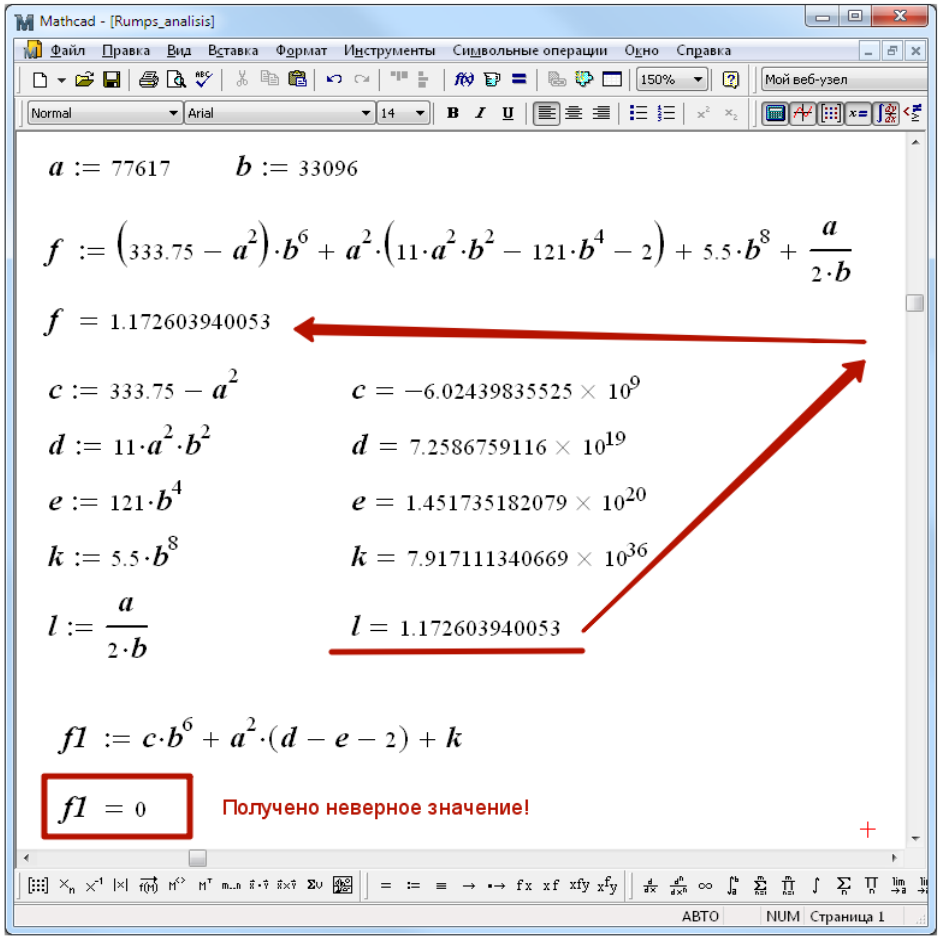


Рисунок 1.1 – Пример расчета полинома Румпа в СКА MachCad 15 с получением неверного результата

Еще один интересный пример, который демонстрирует то, как накопление погрешности влияет на конечный результат вычислений:

$$f(x) = \log_2 x - \frac{\ln x}{\ln 2}.$$

Результаты вычисления функции $f(x)$ в СКА Matlab при значениях x от 0 до 500, представлены на рисунке 1.2. График показывает, что при определенных значениях x, функция $f(x) \neq 0$, хотя очевидно, что в результате всегда получается ноль:

$$f(x) = \log_2 x - \frac{\log_{10} x}{\log_{10} 2} = \log_2 x - \log_2 x = 0.$$

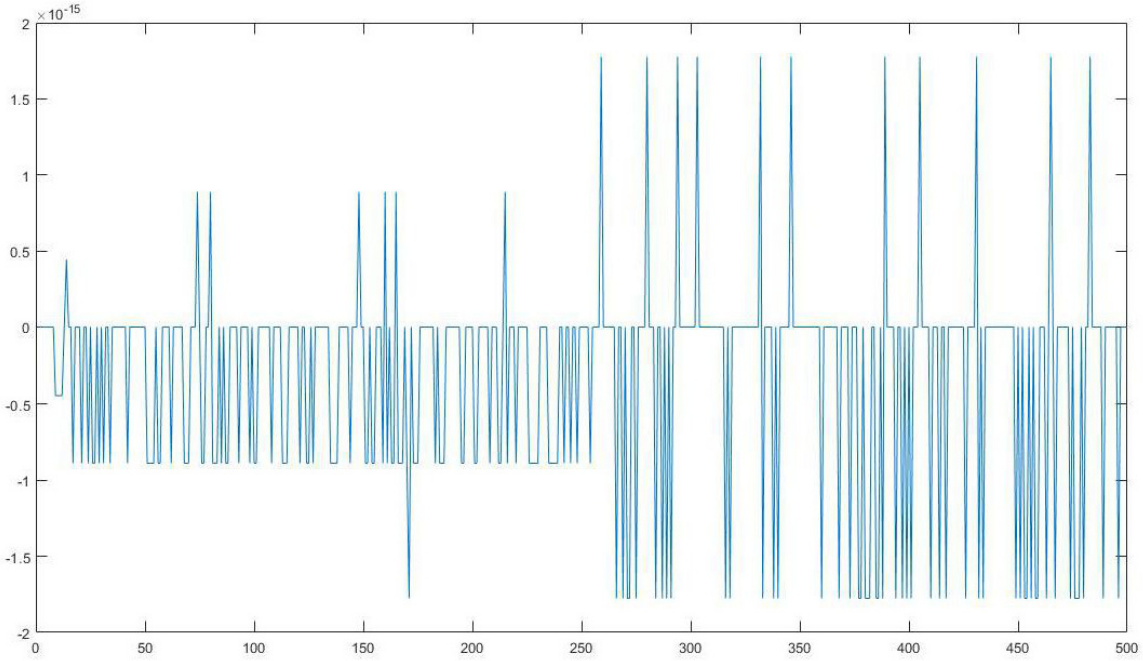


Рисунок 1.2 − Результаты вычисления функции $f(x)$ в СКА Matlab, при значениях x от 0 до 500 в ряде случаев приводят к появлению погрешности

В работе [53] показана ошибочность вычислений при представлении данных в форматах с плавающей запятой. Так, для двух целочисленных векторов x и y со значениями, например

$$x = (10^{15}, 1500, -10^{18}, 10^{20}, 2, -10^{15}),\ y = (10^{15}, 3, 10^{12}, 10^{13}, 222, 10^{18})$$

скалярное произведении равно:

$$x \times y = 10^{30} + 4500 - 10^{30} + 10^{33} + 444 - 10^{33} = 4944.$$

Арифметика чисел с плавающей запятой на любом современном компьютере возвращает для такого произведения нулевое значение:

$$x \times y = 10.0^{15+15}+1500.0\times3.0-10.0^{18+12}+10.0^{20+13}+2.0\times222.0-10.0^{15+18} = 0.$$

Причина этого в столь большой разнице порядков слагаемых, что обычное представление чисел с плавающей запятой не позволяет корректно выполнить вычисление. Полная потеря правильного результата возникает, несмотря на то, что данные используют менее 5% диапазона значений порядка, доступного на большинстве компьютеров.

Таким образом, на современных компьютерах можно получить ошибку вычислений, которая больше самого результирующего числа. При этом не возникает никаких прерываний и сообщений пользователю: со стороны машины вычисления произведены корректно.

Подобные «вычислительные аномалии» получили резонанс и в мировом научном сообществе. В работе [46] указывается, что беспристрастный анализ традиционного подхода к численным вычислениям и соответствующего инструментария приводит к недоопределенности вычислительного алгоритма, который обладает непредсказуемыми свойствами. Директор Института математики в Миннеаполисе, США Дуглас Н. Арнольд не только утверждает, что целый ряд крупнейших аварий с человеческими жертвами и миллиардными убытками всецело обязан нынешней технологией компьютерных вычислений и представлений данных по стандарту IEEE 754, но и приводит конкретные примеры [59]:

- взрыв ракеты «Пэтриот» в Саудовской Аравии 25 февраля 1991, который привел к гибели 28 человек, связан с ошибками округления.
- взрыв ракеты Ariane-5 при ее первом испытании во Французской Гвиане 4 июня 1996 был следствием переполнения числовой сетки компьютера.
- затопление нефтяной платформы 23 августа 1991 в Норвегии, как предполагается, в результате неточного анализа методов конечных элементов (inaccurate finite element analysis).

В работе [48] проф. Ю. П. Петровым выполнен анализ катастроф, причины которых связаны с неточностями методов проектирования и расчета. Помимо анализа многочисленных авиационных катастроф (например, гибель самолета ТУ-154 авиакомпании «Пулково» 22 августа 2006 года над Донецком) также приведена и проанализирована серия аварий зданий и сооружений по всему миру.

1.1.2 Пути возможного повышения эффективности вычислений современного компьютинга

Состояние современной компьютерной арифметики можно существенно улучшить, т. е. сделать ее более «интеллектуальной», путем новых средств и методов аппаратно-программной поддержки. В работе [60, с. 9–10] высказана возможность создания в арифметическом устройстве (или моделирования с помощью программных средств) специального регистра с фиксированной запятой, покрывающего своей разрядностью весь диапазон чисел с плавающей запятой. Возникающая при этом потеря скорости вычислений компенсируется существенным возрастанием их надежности, что также приводит к повышению эффективности компьютерных вычислений. Однако для обеспечения автоматического контроля погрешностей в существующих компьютерных архитектурах машинную арифметику необходимо как минимум обеспечить возможностью направленного округления результата, т. е. округления до ближайшего машинного числа с недостатком или избытком. Данная концепция представления действительных чисел, а также арифметические операции с составленными из них векторами и матрицами позволяют построить машинную интервальную арифметику [34], в которой интервалы представляются в виде компьютерных континуальных объектов и открывают для численного анализа совершенно новую перспективу. В памяти ЭВМ интервал записывается парой чисел в формате с плавающей запятой, определяемых как границы этого интервала. Такой интервал фиксирует все множество вещественных чисел, заключенных между двумя хранимыми в ЭВМ машинными числами.

Арифметические операции с такими парами чисел (границами интервалов-операндов) являются составляющими арифметических операций над интервалами, результатом которых также является пара машинных чисел, представляющая границы результирующего интервала.

Кроме интервального подхода на сегодняшний день сформирован ряд подходов для повышения вычислительной точности, в том числе и направленных на обеспечение достоверных результатов при реализации компьютерных вычислений над числами в форматах с плавающей запятой.

В работе [47] проф. В. Юровицким в предложен способ выражения множества действительных математических чисел эквивалентным множеством метрологических чисел с единой простой метрологической характеристикой – индексом точности. Существенный недостаток данного подхода заключается в том, что определение метрологической характеристики должно осуществляться автоматически при обработке метрологических чисел. В настоящее время отсутствует система ввода-вывода метрологических чисел в компьютер, т. е. номинала числа и его метрологии, а отсутствие единой всеобщей универсальной системы метрологических вычислений говорит об отдаленности заявленной технологии к возможностям современного компьютинга.

В цикле статей [50, 61, 62] проф. Г. Л. Литвиновым и его соавторами рассматриваются универсальные численные алгоритмы, обеспечивающие вычисления с произвольной точностью. Эти алгоритмы основаны на принципах идемпотентной математики, которая в свою очередь направлена на изучение полуколец с идемпотентным сложением. В своих работах авторы представляют приближенную рациональную арифметику с контролируемыми ошибками округления. В частности, в работе [61] утверждается, что: *«Многие алгоритмы не зависят от конкретных математических (алгебраических) операций над данными. В этом случае операции рассматриваются как переменные. Алгоритмы такого рода реализуются обобщенными программами (generic programs), основанными на технике абстрактных типов данных»*. В таких вычислениях в

качестве абстрактных типов данных выступает объект «число», являющийся элементом некоторого «числового» домена – идемпотентного числового кольца.

При этом универсальные алгоритмы, предложенные Л. Г. Литвиновым, не зависят от точности вычислений и конкретного машинного представления чисел, что может вызвать ряд затруднений при их использовании в различных компьютерных архитектурах, ориентированных на вещественную арифметику с плавающей запятой.

В работе [49] российскими учеными Ю. П. Петровым и Л. Ю. Петровым рассмотрены явления, которые в ряде случаев изменяют корректность задач и могут привести к серьезным ошибкам. Продолжением данных исследований является работа [48, с. 32], в которой раскрыты причины многих ошибок и неточностей компьютерных расчетов, связанные со свойствами эквивалентных (равносильных) преобразований. Также описаны методы, позволяющие избежать ошибок, обеспечить достоверность и надежность компьютерных вычислений.

Недостатком данных методов является то, что применение эквивалентных преобразований приводит к анализу и значительной предварительной работе по модификации или полному изменению алгоритмов всего вычислительного процесса. При решении задач на ЭВМ с учетом эквивалентных преобразований, обнаруживается ряд затруднений, связанных с перепрограммированием уже существующих и хорошо зарекомендовавших себя вычислительных алгоритмов.

Проблемой вещественной арифметики в ЭВМ также заняты компании, производящие электронные устройства и компьютерные компоненты, включая микропроцессоры и наборы системной логики. Компания Intel в 2010 году представила расширение системы команд x86 для микропроцессоров Intel и AMD – новую версию SIMD-расширений Advanced Vector Extensions (AVX), – набор улучшенных (по отношению к технологии SSE) и новых инструкций и новую схему кодирования машинных кодов [63]. Расширение AVX добавляет к существующим 128-битным SSE-инструкциям 256-битные инструкции, и в будущем предполагается расширение векторных регистров до 512 или 1024 бит [64, с. 3]. В частности, новая архитектура Intel MIC (Intel Many Integrated Core),

запущенная компанией в 2012 году имеющая кодовое обозначение проекта «Intel Larrabee», уже содержит блок 512-разрядного векторного процессора, способного одновременно обработать 16 чисел одинарной точности с плавающей запятой. Это в четыре и в два раза больше, чем могут обработать на наиболее распространенных процессорах x86 блоки SSE и AVX соответственно [65].

Технология AVX хорошо зарекомендовала себя в плане быстродействия при интенсивных вычислениях с плавающей запятой в мультимедиа программах и научных задачах. Новая архитектура Intel MIC также как и технология AVX увеличивает производительность с вещественными числами и эффективна там, где возможна более высокая степень параллелизма. Однако, в плане точности, увеличение разрядности регистров не устраняет погрешности при вычислениях с вещественными числами, т. к. независимо от точности представления (одинарная, двойная, четверная и т. д.), использование чисел с плавающей запятой формата IEEE 754 может вызвать ряд ошибок компьютерных вычислений [66, с. 196–210]:

1. Ошибки, связанные с точностью представления вещественных чисел в формате IEEE 754. Так как числа с плавающей запятой представляют конечное множество, на которое отображается бесконечное множество вещественных чисел, то исходное число может быть представлено в формате IEEE 754 с ошибкой. При этом величины абсолютных ошибок представления вещественных чисел в формате с плавающей запятой могут быть значительными и при вычислениях приводить к ошибочным результатам.

2. Ошибки, связанные с неправильным приведением типов данных. Согласно стандарту IEEE 754, значения в форматах с плавающей запятой различной точности одного исходного числа обычно не равны друг другу. При создании программных продуктов, оперирующих числами с плавающей запятой не только все вещественные переменные, но и все промежуточные результаты компьютерных вычислений должны быть приведены к одному типу.

3. Ошибки вычислений, вызванные сдвигом мантисс и связанные с потерей точности результата при неполном пересечении мантисс чисел на вещественной оси. Если мантиссы чисел не пересекаются на числовой оси (т. е. сами числа

отличаются более чем на 223 (для формата одинарной точности) и на 252 (для формата двойной точности)), то операции сложения и вычитания между этими числами невозможны.

4. Ошибки, которые возникают при работе с числами, находящимися на границе нормализованного и денормализованного представления чисел. Они связанны с различием в представлении и обработке денормализованных и нормализованных чисел в форматах IEEE 754, что приводит к усложнению цифровых устройств и вычислительных алгоритмов. Также возникает неопределенность переходной зоны, которая заключается в том, что стандарт не определяет конкретного значения границы перехода от максимального денормализованного числа до минимального нормализованного. Точность границы перехода можно задавать до бесконечности, поэтому цифровому устройству или программному продукту может не хватить разрядности для принятия решения, к какому диапазону отнести данное число.

В книге Д. Кнута «Искусство программирования» [67] детально освещен ряд распространенных ловушек, связанных с использованием в программных продуктах арифметики чисел с плавающей запятой. В частности Д. Кнут отмечает: *«Вычисления над числами в формате с плавающей точкой неточны по самой своей природе, и программисту нетрудно столь неудачно организовать их выполнение, что полученные результаты будут почти полностью состоять из «шума»* [67, с. 265].

С точки зрения повышения эффективности компьютерных вычислений представляет интерес идея формирования обобщенного (многомерного) кодо-логического базиса, концепция которого впервые была сформулирована в работе А. Я. Аноприенко [28]. Перспективы использования многомерного кодо-логического базиса затрагивают ряд направлений, ориентированных на практическое развитие возможностей компьютерного моделирования и представления знаний:

- развитие понятия числа в контексте многомерного логического пространства;

- развитие интерфейсных средств в контексте развития кодо-логического базиса;
- эволюция алгоритмического базиса на основе развития кодо-логических средств.

При этом в связи с тесной взаимосвязанностью и взаимообусловленностью компьютерной логики и арифметики, рассматривается именно кодо-логическая эволюция как целостное закономерное явление, определившее в свое время переход от добинарного (прабинарного) компьютинга к современному бинарному. Анализ закономерностей этого перехода позволяет прогнозировать переход в ближайшие десятилетия к постбинарному компьютингу, неизбежность которого обусловлена интенсивным развитием современных компьютерных технологий [68]. В работе [29] в рассмотрение вводятся понятия «тетралогика» и «тетракоды», а также высказывается возможность отображения расширенного кодо-логического базиса как средствами бинарного компьютинга, так и базисами более высоких порядков [69].

Таким образом, повышение эффективности компьютерных вычислений возможно путем объединения двух наиболее перспективных направлений:

1. Интервальный анализ как средство учета ошибок округления компьютерных вычислений, в котором также определены вычисления с плавающей запятой.
2. Переход к расширенному кодо-логическому базису с применением постбинарной логики и постбинарного кодирования для компьютерных вычислений.

1.1.3 Особенности применения интервального анализа для обеспечения достоверных результатов в компьютерных вычислениях

В настоящее время к интервальному представлению факторов неопределенности обращено пристальное внимание инженеров и конструкторов, как к наименее ограничительному и наиболее адекватному описанию начальных

условий при практической постановке инженерных задач. Интервальная неопределенность величины, выраженная своими крайними значениями, может рассматриваться как интервальный параметр, имеющий следующие особенности [14, 70]:

1) любая величина, имеющая интервальную неопределенность, может быть представлена только крайними значениями – границами возможных изменений (либо пределами изменения) этой величины;
2) ширина интервала, которым выражена такая величина, является естественной мерой ее неопределенности (неоднозначности);
3) результатом арифметических операций над величинами, имеющими интервальную неопределенность, также является интервальная неопределенность.

При этом интервальная неопределенность некоторой величины может быть выражена интервальным параметром (называемым также интервалом), который в качестве основного объекта данных является числовым промежутком и не содержит никакой дополнительной информации о самой величине [33]. Так, в контексте интервального анализа отрезок $\boldsymbol{x}$ трактуется как множество возможных значений неизвестной истинной величины, или как ограниченный интервал ее неопределенности:

$$\boldsymbol{x} = [x \mid x_1 \le x \le x_2], \tag{1.2}$$

задаваемый нижней (левой) и верхней (правой) границами x_1 и x_2, причем $x_1, x_2 \in \mathbb{R}$. В различных источниках границы интервала имеют другие обозначения. Например, справедливыми являются тождественности $x_1 \equiv x_- \equiv \underline{x}$ для нижней границы интервала и $x_2 \equiv x_+ \equiv \overline{x}$ – для верхней границы интервала.

Предполагается, что точное значение переменной достоверно находится внутри замкнутого интервала $\boldsymbol{x}$, который принадлежит множеству всех интервалов вещественных чисел $\mathbb{IR}$. При этом для $\boldsymbol{x} \in \mathbb{IR}$ все значения $x \in \mathbb{R}$ считаются равновозможными, следовательно, интервал $\boldsymbol{x}$ не определен никакой вероятностной мерой. Также всякое вещественное число a из $\mathbb{R}$ является

элементом $\mathbb{IR}$, поскольку может быть представлено в виде вырожденного (точечного) интервала $[a; a]$. Следовательно, при $x_1 = x_2 = \boldsymbol{x}$ интервальное число $\boldsymbol{x}$ отождествляется с вещественным числом $x \in \mathbb{R}$, следовательно $\mathbb{R} \subset \mathbb{IR}$.

Использование замкнутых интервалов, в том числе и вырожденных, в качестве числовых аргументов составляет главную идею интервальных вычислений, которая начала интенсивно развиваться в конце XX века в тесной связи с развитием и распространением практических инженерных вычислений. Основы интервального анализа как научной дисциплины были изначально заложены теорией измерений в метрологии, где предполагается, что значение неизвестной величины x характеризуется полученной неточным измерительным прибором недостоверной величины x_0 и известной абсолютной Δ или относительной δ ошибками измерения. Тогда, аналогично (1.2), границы интервала неопределенности $\boldsymbol{x}$ измеряемой величины x_0 выражаются следующим условием:

$$\boldsymbol{x} = [x_1, x_2] = [x_0 - \Delta, x_0 + \Delta] = [x_0 \cdot (1 - \delta), x_0 \cdot (1 + \delta)]. \qquad (1.3)$$

Интервальные вычисления предполагают выполнение унарных и бинарных операций над интервалами. Так, если $\bullet \in \{+, -, \cdot, :\}$ – бинарная операция на множестве $\mathbb{R}$ и $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{IR}$, то

$$\boldsymbol{x} \bullet \boldsymbol{y} = \{x \bullet y \mid x \in \boldsymbol{x}, y \in \boldsymbol{y}\}. \qquad (1.4)$$

Выражение (1.4) определяет бинарную операцию на множестве $\mathbb{IR}$. Если $f(x)$ – непрерывная унарная операция на R, то выражение

$$f(\boldsymbol{x}) = [\min f(x), \max f(x)] \text{ при } x \in \boldsymbol{x} \qquad (1.5)$$

определяет соответствующую ей бинарную операцию также на $\mathbb{IR}$.

Под шириной замкнутого интервала $\boldsymbol{x}$ понимают величину, определяющую разницу его границ (1.6), а под его модулем – максимальную по модулю границу интервала (1.7):

$$\operatorname{wid} \boldsymbol{x} = x_2 - x_1; \qquad (1.6)$$

$$|\boldsymbol{x}| = \max(|x_1|, |x_2|). \qquad (1.7)$$

Важнейшими характеристиками интервала являются его середина (центр), определяемый как

$$\operatorname{mid} \boldsymbol{x} = \frac{1}{2}\left(x_1 + x_2\right), \tag{1.8}$$

и радиус (расстояние от границы интервала до его середины):

$$\operatorname{rad} \boldsymbol{x} = \frac{1}{2}\left(x_2 - x_1\right). \tag{1.9}$$

Отсчет относительно широкого распространения интервальных методов в компьютерных технологиях ведется с первого международного симпозиума по интервальному анализу, прошедшего в Великобритании в январе 1968 года [35]. Первая монография, полностью посвященная интервальному анализу, была опубликована Р. Э. Муром в 1966 году [36], в которой практически впервые были последовательно изложены основы нового направления в вычислительной математике.

Ранее в работе [70] в 1931 году была предложена арифметика для вычислений с множествами чисел. В 1951 году введен в научный оборот специальный случай замкнутых интервалов, обусловленный необходимостью учета погрешностей в численном анализе [71,72]. В 1956–1958 годах в работах [73, с. 253–259] и [74] была предложена классическая интервальная арифметика и ее приложения. Кроме того, в работе [75, с. 167] были заложены основы интервального алгебраического формализма и даны весьма нетривиальные примеры применений новой техники, к примеру, в численном решении алгебраических уравнений и задачи Коши для обыкновенных дифференциальных уравнений.

В 1962 году в одном из первых выпусков «Сибирского математического журнала» появилась статья Л. В. Канторовича [45, с. 701], обозначившего эту тематику как одну из приоритетных для активно набирающей обороты вычислительной науки. К настоящему времени разработаны различные приемы интервальных вычислений [37, 41] и множество пакетов прикладных программ и

алгоритмических макроязыков, реализующих элементы интервального анализа на машинном уровне для нескольких типов ЭВМ [76, 77].

На современных ЭВМ интервальный тип данных реализуется, как правило, с помощью представления интервала в виде пары чисел – одного для левого конца интервала, а другого для правого. При этом существующее аппаратное обеспечение, в частности, арифметика чисел с плавающей запятой, используется без каких-либо изменений, так как корректность получающейся интервальной арифметики может быть обеспечена направленными округлениями. Например, там, где в задачах внешнего интервального оценивания в процессе вычислений требуется округление результата, нижняя граница интервала должна округляться до ближайшего машинного числа с недостатком, а верхняя граница интервала – до ближайшего машинного числа с избытком, что не противоречит равенству (1.3). Таким образом, даже неизбежные ошибки округления при вычислениях с плавающей запятой будут строго и систематически учитываться в процессе выполнения интервальной программы. В качестве примера, на рисунке 1.3 показано, как иррациональные числа $\sqrt{2}$ и π в различных числовых шкалах представлены в виде различных числовых промежутков (интервалов), причем наглядно проиллюстрировано изменение ширины (1.6) этих интервалов. Интервалы чисел, представленных в вещественном формате (шкала floats) являются достаточными, так как в границы интервалов включены и ошибки округления исходных иррациональных чисел: $\sqrt{2} \in [1.25, 1.5]$; $\pi \in [3.0, 3.5]$.

Данный пример отражает основные положения интервального подхода к вычислениям на ЭВМ: исходные данные и промежуточные результаты представляются граничными значениями, над которыми и производятся все арифметические операции согласно соотношению (1.4). При этом сами операции определяются таким образом, что результат соответствующей точной операции обязательно лежит внутри вычисляемых границ.

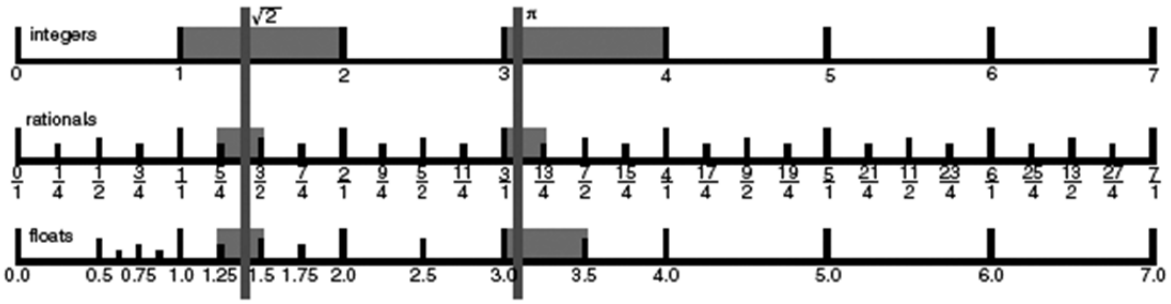


Рисунок 1.3 – Представление иррациональных чисел в виде интервалов (integers – целые числа, rationals – рациональные числа, floats – вещественные числа) [78, с. 485]

Иллюстрацией интервального подхода могут служить следующие правила выполнения операций вещественной интервальной арифметики:

$$[x_1, x_2] + [x_3, x_4] = [x_1 + x_3, x_2 + x_4]; \tag{1.10}$$

$$[x_1, x_2] - [x_3, x_4] = [x_1 - x_4, x_2 - x_3]; \tag{1.11}$$

$$[x_1, x_2] \times [x_3, x_4] = [\min(x_1x_3, x_1x_4, x_2x_3, x_2x_4), \max(x_1x_3, x_1x_4, x_2x_3, x_2x_4)]; \tag{1.12}$$

$$[x_1, x_2] / [x_3, x_4] = \left[\min\left(\frac{x_1}{x_3}, \frac{x_1}{x_4}, \frac{x_2}{x_3}, \frac{x_2}{x_4}\right), \max\left(\frac{x_1}{x_3}, \frac{x_1}{x_4}, \frac{x_2}{x_3}, \frac{x_2}{x_4}\right) \right], \tag{1.13}$$

где $0 \notin [x_3, x_4]$.

Для оценки ширины wid результирующего интервала справедливы следующие равенства:

$$\text{wid}\,(\boldsymbol{x}_I \pm \boldsymbol{x}_J) = \text{wid}\,\boldsymbol{x}_I + \text{wid}\,\boldsymbol{x}_J; \tag{1.14}$$

$$\text{wid}\,(\boldsymbol{x}_I \times \boldsymbol{x}_J) \geq \max\,\{|\boldsymbol{x}_I| \times \text{wid}\,\boldsymbol{x}_I,\ |\boldsymbol{x}_J| \times \text{wid}\,\boldsymbol{x}_J\}; \tag{1.15}$$

$$\text{wid}\,(\boldsymbol{x}_I / \boldsymbol{x}_J) \leq \{|\boldsymbol{x}_I| \times \text{wid}\,\boldsymbol{x}_I,\ |\boldsymbol{x}_J| \times \text{wid}\,\boldsymbol{x}_J\}. \tag{1.16}$$

Равенства (1.10–1.13), и, в некоторой степени, равенства (1.14–1.16) показывают, что арифметические операции над интервалами требуют больше машинного времени, чем аналогичные операции над вещественными числами. В частности, проведенные в [79] исследования по оценке временных затрат для

реализации арифметических операций с вещественными числами в СКА Mathematica при использовании традиционных и интервальных вычислений показали, что применение последних позволяют успешно решать многие практические задачи, но при этом машинное время увеличивается примерно на порядок. Кроме того, операции интервального вычитания и деления выполняются дольше интервальных операций сложения и умножения соответственно. Также проведение интервальных вычислений требует значительных процессорных ресурсов и объема памяти (по сравнению с традиционными вычислениями). Однако ценность интервального подхода с точки зрения повышения эффективности компьютерных вычислений заключается в том, что применение интервального анализа в целом позволяет получать наиболее достоверные решения исходных задач, учитывающие возможные диапазоны изменения исходных и вычисляемых значений.

1.2 Зарождение и развитие идей постбинарного компьютинга

В работе [80, с. 5] сформулировано обобщенное понятие постбинарного компьютинга. Это информационные и вычислительные технологии будущего, обеспечивающие переход на качественно новый уровень в представлении, хранении и обработке информации. Неизбежность такого перехода определяется всем ходом кодо-логической эволюции в процессе развития технологий работы человека с информацией вообще и с количественной информацией в частности.

Идеи постбинарного компьютинга постепенно начали формироваться в рамках различных исследований и разработок на протяжении всего XX века: это и работы начала века в области неклассической логики Яна Лукасевича и Николая Васильева, это и троичный компьютер «Сетунь», разработанный в конце 1950-х годов в вычислительном центре Московского государственного университета под руководством Николая Брусенцова, и множество прочих теоретических и практических работ, из которых к рубежу тысячелетий начал проясняться образ постбинарного будущего. Именно на этом фундаменте сформировалось видение

будущего компьютинга, как неизбежное проявление долговременных тенденций и закономерностей развития информационных и компьютерных технологий.

В 90-е годы начали публиковаться первые результаты исследований в области постбинарного компьютинга [29, 81, 82, 83]. Зарождение концепции постбинарного компьютинга основывается на развитии идей расширенного кодо-логического базиса, кодо-логической эволюции и квазигенетического кодо-логического компьютинга. В то же время, расширение кодо-логического базиса выражается в использовании новых логических и цифровых значений, что само по себе не обязательно ведет к кардинальным изменениям в средствах и методах компьютинга. Но при более детальном рассмотрении выясняется, что постбинарный компьютинг позволяет преодолеть целый комплекс недостатков, присущих обычному бинарному (двоичному) компьютингу, и открыть целый комплекс возможностей развития компьютерных технологий.

В настоящее время компьютеры оперируют с числами различной разрядности и, соответственно, точности. Например, для чисел с плавающей запятой речь может идти об одинарной, двойной или четверной точности. Программист в процессе решения задачи может выбрать наиболее приемлемый уровень точности. Возможен и автоматический выбор точности в процессе реализации определенных вычислительных алгоритмов. Но при этом практически полностью игнорируется информация об изначальной точности используемых данных, которые могут быть получены различными способами (например, путем реальных измерений с ограниченной точностью, путем экспертных оценок с еще более ограниченной точностью и т. д.). Это во многих случаях приводит к явно завышенным по сравнению с реальными оценкам точности получаемых результатов и/или к существенным искажениям этих результатов, как правило, остающимся в такой ситуации незамеченными. Постбинарный компьютинг позволяет ввести тотальный контроль текущей точности всех количественных значений, что представляется абсолютно необходимым в контексте наблюдаемых в настоящее время тенденций цифровизации.

Также переход от кодирования (на уровне одиночных компьютерных числовых значений) точечных цифровых значений на числовой оси к представлению одним числовым кодом множественных и/или вероятностных значений позволит существенно более компактно и адекватно кодировать информацию об объектах и процессах реального мира.

Далее, с существенным повышением гибкости количественного представления информации значительно повышаются возможности представления и обработки логических значений, что открывает новые возможности в компьютерном моделировании реальных процессов и построении систем искусственного интеллекта. Не отменяя существующие технологии бинарного компьютинга, а значительно расширяя и дополняя их, постбинарный компьютинг позволяет существенно расширить и обогатить интеллектуальный инструментарий накопления, хранения и обработки информации, открывая новые возможности не только в информационных технологиях, но и в целом в осмыслении и понимании реального мира.

1.2.1 Анализ и интерпретация расширения логического пространства

Исследование особенностей развития расширенного логического пространства представляется актуальным в связи с тем, что понимание основных закономерностей современного бинарного этапа позволит более эффективно реализовать переход к постбинарному этапу в развитии компьютерных технологий. Первые элементы постбинарного компьютинга начали формироваться уже на протяжении XX века. В частности, в 90-е годы как самостоятельное научное направление оформилась новая комплексная дисциплина, известная в настоящее время под названием «вычислительный интеллект», которая должна была стать наиболее перспективной альтернативой искусственному интеллекту. Одной из основных особенностей вычислительного интеллекта явилась его ориентация на «мягкие вычисления» («soft computing»),

концептуально описанные Л. Заде наряду с определением понятия вычислительного интеллекта [84].

Предлагаемая в [29, с. 39] концепция расширенного логического пространства актуальна, во-первых, для обобщения и систематизации уже имеющихся в этой области результатов, а во-вторых – для обеспечения возможности синтеза новых эффективных методов и средств вычислений. Основная идея данной концепции базируется на гипотезе о множественности эволюционирующих кодо-логических форм и методов человеческого мышления.

Расширенное двумерное логическое пространство (Рисунок 1.4) порождено базисом, состоящим из ортонормированной системы векторов «Истина» (обозначается как T – True или Y – «Yes») и «Ложь» (F – False или N – «No») с положительной и отрицательной полуосями [81].

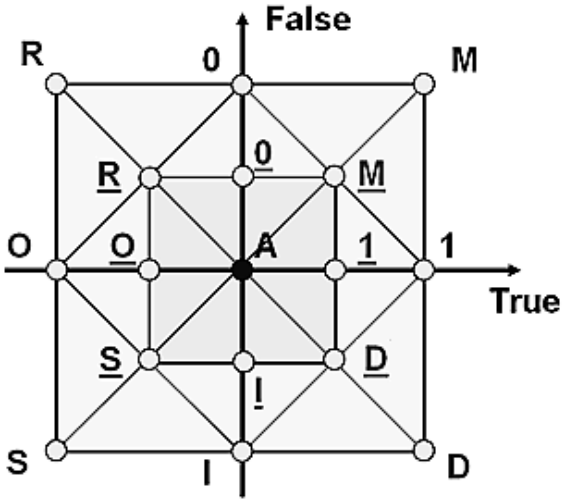


Рисунок 1.4 – Расширенное двумерное логическое пространство

Логические значения при этом могут задаваться либо соответствующими координатами, либо фиксацией характерных точек, определяющих логические состояния: 1 и 0 – «истина» и «ложь» классической логики; A – абсолютная неопределенность, неизвестность; M – множественность, многозначность («истина» и «ложь» одновременно); S – симметричность (инверсная многозначность, отражение M относительно точки A); I и O – инверсные

«истина» и «ложь»; D и R – формы многозначности, по-разному комбинирующие свойства значений M и S.

При этом каждой точке, характеризующей перечисленные выше логические состояния, поставлена в соответствие точка (обозначается аналогичным символом, но с подчеркиванием), значения которой может ассоциироваться с дробностью, половинчатостью, равновероятностью. Например, значение M предполагает равновероятность состояний «истина» и «ложь».

В двумерном логическом пространстве могут быть построены различные логические системы, отличающиеся количеством используемых логических значений. Возможные логические системы представляют собой множество L_K, где K есть количество используемых логических значений или порядок логики, т. е. $L_K = \{x_1, x_2, ..., x_K\}$. При этом логическая система определяется как множеством логических значений L_K, так и множеством логических функций $F(L_K)$.

Аналогично тому, как бинарная логика является основой двоичной системы счисления, на базе вышеперечисленных логических систем могут быть построены соответствующие системы кодирования количественной информации. Все вводимые системы кодирования рассматриваются на машинном уровне, т. е. на уровне двоичной системы счисления, когда кодовый алфавит однозначно совпадает с алфавитом соответствующей логической системы. Системы кодирования при этом задаются аналогично соответствующим логическим системам: $C_K = \{x_1, x_2, ..., x_K\}$.

Перечисленные логические и кодовые системы в совокупности образуют расширенный (обобщенный) кодо-логический базис.

В вычислительной технике возможность и необходимость выхода за пределы одномерного логического пространства впервые была достаточно четко декларирована в 1976 году в работах «Как нужно рассуждать компьютеру» и «Об одной полезной четырехзначной логике» [22], в которых была предложена четырехзначная логика со следующими значениями истинности: T – «только

Истина» (True); F – «только Ложь» (False); N – «ни Истины, ни Лжи» (None); B – «и Истина и Ложь» (Both). Необходимость четырехзначной логики обосновывалась тем, что входные данные могут поступать в компьютер из различных независимых источников, что может привести к достаточно типичной ситуации: появлению противоречивой информации. Предложенная логика рассматривалась как средство практического преодоления такой ситуации.

Таким образом, логика L_4 (тетралогика) может рассматриваться как основа для построения эффективных систем кодирования количественной информации, обладающих по сравнению с традиционной логикой рядом качественных преимуществ.

Основанная на логике L_4 система кодирования C_4 (тетракод) открывает принципиально новые возможности для развития математики и новых вычислительных средств, т. к. представление чисел в виде тетракодов дает возможность гибкого задания не только фиксированных значений на числовой оси, но и множество значений одновременно. По сравнению с обычным бинарным кодом количество бит, требуемых для кодирования чисел, возрастает в $\log_2 n$ раз, где $n = 4$ – количество возможных значений в каждом разряде числа при его машинном представлении. Следовательно, для представления тетракодов требуется удвоенная длина кодовых слов. Однако повышение степени информативности получаемых за счет этого кодов вполне оправдывает увеличение затрат на кодирование. Экспериментально, в частности, доказана эффективность использования тетракодов для представления различных произвольных структур на примере кодирования изображений [82, 83].

Результаты исследований по многомерному логическому пространству были впервые представлены в англоязычном информационном пространстве в 1997 году: доклады на международной конференции по моделированию в Стамбуле [31] и международном конгрессе по научным вычислениям, моделированию и прикладной математике в Берлине [32].

В Донецком национальном техническом университете по результатам исследований в области интервальных вычислений в 2010–2011 годах были опубликованы работы [68, 85, 86, 87, 88], в которых рассматриваются особенности реализации интервальных вычислений в контексте постбинарного компьютинга. В частности, в работе [87] был предложен способ кодирования границ интервала одним значением тетракода $C_4 = \{0, \mathrm{A}, \mathrm{M}, 1\}$, причем разряды множественности (М) являются определяющими для позиционирования значений границ интервала на числовой оси, а разряды неопределенности (А) – уточняющими позицию этих чисел. Варьируя количеством разрядов М и А, можно добиться представление границ интервала с необходимой точностью. В работах [89, 90] рассмотрены способы решений интервальных полиномиальных функций с контролем точности результирующего интервала. Предложены методы замены и дифференцирования, при использовании которых отбрасываются все «побочные» значения, вызванные чрезмерным расширением интервального результата при вычислениях полиномов с интервальными коэффициентами. В работах [57, 91, 92] предложены специальные форматы вещественных чисел различной точности, способные хранить в одном поле числа значения обеих границ интервала. Данный формат является одним из способов хранения данных, которые в дальнейшем планируется использовать при моделировании кардинально новых вычислительных алгоритмов и архитектур на базе существующих компьютерных систем.

1.2.2 Закономерности, определяющие условия перехода к постбинарному компьютингу

Проведенный в работе [80, с. 48–81] анализ показывает, что с 1960-х годов наблюдается устойчивый рост числа программируемых устройств в виде компьютерных систем самого различного назначения и масштаба. Скорость этого роста примерно десятикратная за десятилетие. В результате уже в 2014 году общее количество используемых в мире программируемых устройств превысило

все население земного шара. При этом все свидетельствует о том, что процесс насыщения техносферы компьютерными системами и в обозримом будущем будет происходить такими же темпами, в основном за счет роста и развития инфраструктуры «Интернета вещей». В следующем десятилетии на каждого жителя планеты будет приходиться порядка 10-ти программируемых устройств.

Такое массовое использование программируемых систем, в том числе и жизненно важных программных продуктов, например, для мониторинга человеческого организма, предполагает резкое повышение требований к надежности и достоверности информационно-компьютерного обеспечения таких устройств. И именно с целью удовлетворения этих требований неизбежно потребуется переход к постбинарному компьютингу, где текущий контроль точности и надежности данных и процессов их обработки может осуществляться перманентно уже на уровне представления информации в компьютере.

Неизбежный при этом рост аппаратных затрат на реализацию систем хранения и обработки данных выглядит совсем незначительным на фоне общего роста сложности компьютерных систем, что в первую очередь описывается классическим законом Мура, предполагающим постоянный, начиная с 1960-х годов, рост числа активных элементов (транзисторов) в интегральных микросхемах. И здесь также универсальным ответом на все новые вызовы может быть переход к постбинарному компьютингу, в котором, кстати, неопределенность является одним из естественных свойств данных.

Для обеспечения дальнейшего роста производительности такими же темпами в компьютерных системах осуществляется переход к массовому параллелизму вычислительных процессов. В постбинарном компьютинге за счет возможности задания множественности уже на уровне представления данных параллелизм обработки является неотъемлемым и естественным.

Наиболее вероятный сценарий перехода к постбинарному компьютингу видится следующим: первые постбинарные компьютерные системы на уровне аппаратуры будут реализованы при разработке новых классов устройств

наноуровня, так более ранние и масштабные классы систем довольно инерционны в своем развитии [93].

1.3 Выводы по первой главе

Таким образом, рассмотренные в данной главе сведения о текущем состоянии современных компьютерных вычислений, с учетом получивших в последнее время развитие новых направлений, позволяют заключить следующее:

1. В настоящее время созрели все предпосылки для существенной модификации всей системы компьютерных вычислений с целью повышения ее надежности и адекватности современным требованиям. При этом речь идет о дальнейшем развитии как логической, так и вычислительной составляющей современного компьютинга.

2. Существующая модель вычислений с плавающей запятой не предназначена ни для адекватного представления исходных значений, ни для отслеживания вычислительных ошибок. В связи с этим постепенно усиливается тенденция к переходу от точечных значений к интервальным.

3. При повышении эффективности компьютерных вычислений интервальные методы могут выступать в качестве лишь вспомогательных средств для решения задач, неинтервальных по своей природе, поскольку использование интервальных вычислений «в чистом виде» предполагает возникновение неопределенности и неоднозначности в самом начале, являясь при этом неотъемлемой частью постановки задачи.

4. Применение предложенных Ю. П. Петровым эквивалентных преобразований приводит к анализу и значительной предварительной работе по модификации или полному изменению алгоритмов всего вычислительного процесса. При решении задач на ЭВМ с учетом эквивалентных преобразований, обнаруживается ряд затруднений, связанных с перепрограммированием уже существующих и хорошо зарекомендовавших себя вычислительных алгоритмов,

что не может являться ключевым фактором для повышения эффективности компьютерных вычислений.

5. Предложенные в последнее время технологии ведущих фирм-производителей компьютерных компонентов хотя и направлены на улучшения вычислений с плавающей запятой, но напрямую не связаны с повышением точности. Например, улучшения с использованием технологии AVX фирмы Intel касаются степени параллелизма, системы кодирования и увеличения ширины векторных регистров SIMD со 128 (XMM) до 256 бит (YMM) без наращивания разрядности используемых форматов чисел с плавающей запятой одинарной и двойной точности.

6. Возможность повышения эффективности компьютерных вычислений средствами и методами интервального анализа открывается только при переходе к расширенному кодо-логическому базису. При этом с применением тетралогики и тетракодирования особо актуальной представляется разработка такой модификации вычислений с плавающей запятой, которая позволила бы исключить потерю точности (или информации о ней) при формировании входных, промежуточных и результирующих значений.

ГЛАВА 2
ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ ТЕТРАЛОГИКИ

2.1. Основные тенденции перехода от двоичной логики к тетралогике

Так как логика служит одним из инструментов практически любой науки, то с появлением и широким распространением вычислительной техники логика оказала значительное влияние на развитие искусственного (а позже – и вычислительного) интеллекта.

Классическая булева логика основана всего на двух логических состояниях: истина (англ. false) или логическая единица; ложь (англ. true) или логический нуль. Такая логика из-за простоты реализации получила широкое распространение в вычислительной технике и более точное название: двоичная (двузначная) логика. Считается, что двоичная логика в достаточной степени способна выполнять основную функцию классической логики: исследование того, как из одних утверждений можно выводить другие. Но реальное мышление не сводится к манипуляциям только двумя логическими значениями, поскольку часто приходится иметь дело с противоречивой информацией, например, поступившей от нескольких независимых источников. Использующий двузначную логику компьютер не является достаточно совершенным устройством, которое, столкнувшись с противоречием, было бы способно сделать нечто большее, чем просто зафиксировать его существование. В частности, Н. Белнап в 1976 году высказал предположение, что совершенное устройство должно обладать некоторой стратегией, с тем чтобы, обнаружив противоречивость определенных представлений, иметь возможность отказаться от них [22]. Несмотря на постоянное увеличение вычислительной мощности современных компьютерных систем, существующие логические основы их работы не позволяют в должной степени приблизить искусственный интеллект к человеческому, что делает практически недостижимым одно из требований к

искусственному интеллекту: выполнение функций (в частности, творческих), которые традиционно считаются прерогативой человека [94]. Поэтому современное состояние логического аппарата компьютерных технологий представляет собой рубеж, преодоление которого положит начало созданию компьютерной техники нового поколения, в качестве обобщающего названия для которой целесообразным представляется использование термина «постбинарный компьютинг» [28, 30, 31, 32].

Необходимым условием при построении логической системы является выполнение главной задачи логики, которая базируется на двух ключевых моментах [30]:

- соблюдение «правильных рассуждений» – отработка механизма перехода к правдивым выходным заключениям исходя из входных предпосылок;
- получение «истинного знания» о предмете размышления для детальной проработки нюансов изучаемых явлений и их соотношений друг с другом.

Двоичная логика и основанные на ней системы счисления представляют собой логическую систему, которая по своей сути является одномерной, поскольку строится в пределах оси, соединяющей логические «0» и «1». Такая одномерная логическая система не единственна и не достаточна, однако она является одним из наиболее значимых элементов современного интеллектуального инструментария. В целом можно констатировать, что XX век ознаменовался прогрессирующим нарастанием протеста против двузначности, в частности:

- введение трехзначной логики Я. Лукасесвича, в которой впервые появилось третье истинностное значение, интерпретирующееся как «безразлично», а в поздних редакциях – как «возможно» [17, 95, 96];
- переход Я. Лукасевичем к троичной модальной логике, в которой к уже имеющимся логическим функциям добавляются модальные операторы «возможность», «необходимость» и «случайность» [97];

- появление «логики бессмысленности» Д. А. Бочвара [18], в которой третье истинностное значение предлагается интерпретировать не столько как промежуточное между истиной и ложью, сколько как парадоксальное значение или даже как «бессмыслица». При этом Бочвар, в отличие от Лукасевича, сформировал два типа функций – внутренние и внешние (например, внутреннее отрицание, внешнее утверждение), а также новые по своей логической природе функции, например «бессодержательность», «неверность» [97, с. 120–122];
- введение трехзначных регулярных логик Клини – сильной логики и двух промежуточных регулярных логик, в одной из которых отрицается коммутативность дизъюнкции и конъюнкции [19, 20];
- разработка Н. П. Брусенцовым троичного ферритодиодного элемента и создание на его базе троичной ЭВМ «Сетунь» в вычислительном центре МГУ [98], а также предлагаемые им новые подходы к формированию многозначных логик [99, 100, 101];
- описание четырехзначной логики Белнапа, в которой используются четыре истинностных значения («говорит только истину», «говорит только ложь», «не говорит ни истины, ни лжи» и «говорит и истину и ложь»), для которых Белнап определил три базовые операции: отрицание, конъюнкцию и дизъюнкцию [102, 103].
- появление нечеткой логики Л. Заде, справедливо квалифицируемой «как вызов, брошенный европейской культуре с ее дихотомическим видением мира в жестко разграничиваемой системе понятий» [14, 15, 16];
- наряду с появляющимся разнообразием конечнозначных логик, обозначился переход к бесконечнозначным логикам, среди которых наиболее известная логика Поста.

Другими важными составляющими являются как некоторые более ранние формы мышления и представления количественной информации, так и множество перспективных, существующих пока в не полностью оформившемся виде, но все

же обладающих значительным информационным потенциалом. К одной из таких перспективных составляющих можно отнести рассмотренное ранее двумерное логическое пространство (Рисунок 1.4). Таким образом, введение новых логических значений позволяет значительно расширить возможности формализованной логической оценки разнообразных реальных процессов и ситуаций, что является существенным шагом к «очеловечиванию» машинной логики. В качестве наиболее перспективного варианта, из всех возможных сочетаний логических состояний (или высказываний) в двумерном логическом пространстве, рассматриваются четыре: классические «истина» (1) и «ложь» (0); значения неопределенности (А) и множественности (М). Данный вариант тетралогики (Рисунок 2.1) можно представить в виде множества четырех состояний $L_4 = \{0, A, M, 1\}$.

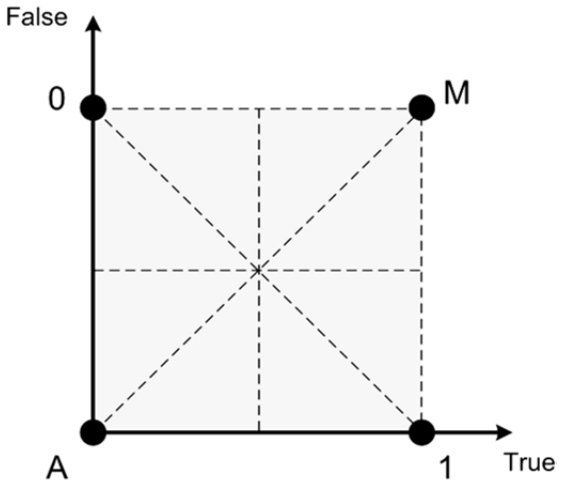


Рисунок 2.1 – Двумерное логическое пространство с указанными логическими значениями рассматриваемого варианта тетралогики

Следует отметить, что в тетралогике логические ноль и единица выступают в качестве однозначных логических состояний (однозначно представимых, т. е. фактически повторяющих логические состояния двоичной логики), а множественность и неопределенность – в качестве неоднозначных логических состояний.

Таким образом, тетралогика в простейшем варианте является конечнозначной и, согласно теории функциональных систем [44], может быть описана классом *k*-значных функций при $k = 4$. При этом имеет место запись так называемой тетрафункции $Tr^n \to T$, где T = {0, A, M, 1} – выбранное множество тетралогики, $n \in \mathbb{Z}$, $n \geq 0$ – арность или местность функции. В данном представлении тетрафункции, элементы Tr^n можно назвать векторами тетралогики (тетравекторами). В случае $n = 0$ тетрафункция превращается в константу T – одно из представленных состояний тетритов.

В *k*-значной логике максимальное количество возможных *n*-местных логических операций определяется как число функций N_k^n:

$$N_k^n = k^{(k^n)\cdot m}, \tag{2.1}$$

где k – число знаков (основание системы счисления), n – число аргументов (входов), а m – число выходов.

Таким образом, согласно (2.1), в тетралогике определено $N_4^n = 4^{(4^n)\cdot m}$ простейших *n*-арных тетрафункций с *m*-арным результатом (выходом). При этом нульарные (n = 0), унарные (n = 1) и бинарные (n = 2) операции тетралогики определяются соответствующими тетрафункциями.

В последующих разделах данной главы определены возможности выполнения основных операций тетралогики, в частности реализации ряда унарных и бинарных поразрядных логических операций с тетритами. При этом проанализированы свойства операций тетралогики на соблюдение законов традиционной алгебры логики.

2.2 Нульарные и унарные логические операции

Согласно (2.1) в тетралогике могут быть определены $N_4^0 = 4^{(4^0)\cdot 1} = 4^1 = 4$ простейшие нульарные тетрафункции T_i (i = 1÷4), фактически являющиеся логическими константами:

- логический тождественный ноль, выражающий ложность события или логического состояния: $T_1 = 0$;
- неопределенность A, выражающая ни ложность ни истинность события или логического состояния, и которую можно выразить математическим соотношением совокупности логических нуля и единицы: $T_2 = \begin{bmatrix} 0, \\ 1 \end{bmatrix} = \mathrm{A}$;
- множественность M, выражающая одновременно и ложность и истинность события или логического состояния и которую можно выразить математическим соотношением системы логических нуля и единицы: $T_3 = \begin{cases} 0, \\ 1 \end{cases} = \mathrm{M}$;
- логическая тождественная единица, выражающая истинность события или логического состояния $T_4 = 1$.

Одноместные (унарные) функции тетралогики определены в количестве $N_4^1 = (4)^{4^1 \cdot 1} = 4^4 = 256$ (в отличие от 4-х и 27-ми унарных функций двоичной и троичной логики). Количество возможных унарных функций тетралогики также равно числу размещений с повторениями $\overline{A}_n^k$ при $k = n = 4$: $\overline{A}_n^k = n^k = 4^4 = 256$.

В таблице 2.1 приведены названия и обозначения 16-ти базовых унарных тетрафункций, а на рисунках А.1 и А.2 приведена графическая интерпретация некоторых базовых унарных тетрафункций для логического состояния x, определенного на двумерном логическом пространстве как множество $x \in \mathrm{L}_4$.

Таблица 2.1 – Базовые унарные функции тетралогики

x				*Название тетрафункции*	*Обозначение*
A	1	M	0		
Tr(x)					
0	0	0	0	тождественный ноль	O(*x*)=0
0	1	M	0	минимизация неопределенности	MIN_A(*x*)
0	A	1	M	циклический сдвиг (поворот, вращение) назад на 1 шаг (1/4 оборота) или вперед на 3 шага (3/4 оборота)	SHIFT_1B(*x*), SHIFT_3F(*x*)
0	M	1	A	вероятностная инверсия по оси «False»	SWAP_FL(*x*)
1	1	1	1	тождественная единица	I(*x*)=1
1	1	M	0	максимизация неопределенности	MAX_A(*x*)
1	A	0	M	вероятностная инверсия по оси «True»	SWAP_TR(*x*)
1	M	0	A	циклический сдвиг (поворот, вращение) вперед на 1 шаг (1/4 оборота) или назад на 3 шага (3/4 оборота)	SHIFT_1F(*x*), SHIFT_3B(*x*)
A	0	M	1	инверсия с фиксацией неопределенности и множественности (классическая инверсия)	SWAP_A/M(*x*)
A	1	0	0	минимизация множественности	MIN_M(*x*)
A	1	1	0	максимизация множественности	MAX_M(*x*)
A	1	M	0	тождественная функция, повторитель	*x*, SHIFT_0B(*x*), SHIFT_0F(*x*),
A	A	A	A	абсолютная неопределенность	A(*x*)=A
M	0	A	1	постбинарная (полная) инверсия или циклический сдвиг (поворот, вращение) вперед на 2 шага (1/2 оборота) или назад на 2 шага (1/2 оборота)	$\overline{x}$, #*x*, SHIFT_2F(*x*), SHIFT_2B(*x*)
M	1	A	0	инверсия с фиксацией нуля и единицы	SWAP_0/1(*x*)
M	M	M	M	абсолютная множественность	M(*x*)=M

2.2.1 Свойства логических операций инверсной группы

Унарные операции, которые могут быть получены путем вращения двумерного логического пространства по осям симметрии (обмен или инвертирование местоположения пар значений логического пространства), будем называть операциями инверсной группы – SWAP.

Для унарных логических операций инверсной группы (Рисунок А.1) справедлив положенный в основу классической логики закон двойного отрицания [104], который в контексте тетралогики представляет явление, расширяющее не только концепцию двойного отрицания как такового, но и принадлежность его к определенному виду. Соответствующая речевая конструкция данного закона может быть, например, расширена до следующего выражения: *«если известно, как именно неверно, что неверно x, то аналогичным образом известно, что верно x»*.

В классической логике двойное отрицание высказывания x является следствием суждения x, поэтому в рамках тетралогики имеет место тавтология:

$$x \rightarrow \text{SWAP\_}\xi\ (\text{SWAP\_}\xi\ (x)) \quad \text{при} \quad \xi = \{\text{A/M, 0/1, FL, TR}\}.$$

Обратное утверждение

$$\text{SWAP\_}\xi(\text{SWAP\_}\xi(x)) \rightarrow x \quad \text{при} \quad \xi = \{\text{A/M, 0/1, FL, TR}\}$$

также не противоречит закону двойного отрицания в классической логике. Здесь ξ определяет один из видов отрицания (инверсии) и выражает первую часть адаптированной под тетралогику речевой конструкции: *«если известно, как именно…»*.

В обобщенной записи закона двойного отрицания инверсных операций тетралогики можно воспользоваться свойствами математической равнозначности:

$$x \equiv \text{SWAP\_}\xi(\text{SWAP\_}\Psi(x));$$
$$\xi = \{\text{A/M, 0/1, FL, TR}\}.$$

2.2.2 Свойства логических операций сдвиговой группы

Для унарных логических операций сдвиговой группы определены направления сдвига. Так, под словом «назад» (сдвиговая операция SHIFT_<*i*>B, B от англ. *Backward* – назад) подразумевается направление сдвига, совпадающее с направлением оси «False» двумерного логического пространства (Приложение А, Рисунок А.2). Аналогичным образом под словом «вперед» (сдвиговая операция

SHIFT_<*i*>F, F от англ. *Forward* – вперед) – направление сдвига, совпадающее с направлением оси «True» [68, 105, 106]. Целое положительное число <*i*> ($i \geq 0$) определяет количество сдвигов (или «поворотов» условной плоскости в концепции двумерного логического пространства).

На рисунке А.2 (Приложение А) пунктирными линиями показаны соответствия при выполнении сдвиговых унарных операций тетралогики. При $i = 0$ сдвиговые унарные операции SHIFT_<*i*>ς при ς = {B, F} идентичны и соответствуют тождественной функции, т. е. являются повторителями: $x \to \text{SHIFT\_0B}(x) \to x$; $x \to \text{SHIFT\_0F}(x) \to x$.

Очевидно, что $x \equiv \text{SHIFT\_0B}(x)$; $x \equiv \text{SHIFT\_0F}(x)$, откуда следует соответствие $\text{SHIFT\_0B}(x) \equiv \text{SHIFT\_0F}(x)$.

Аналогичным образом прослеживаются следующие идентичности (равнозначности):

$$\text{SHIFT\_3B}(x) \equiv \text{SHIFT\_1F}(x);$$
$$\text{SHIFT\_1B}(x) \equiv \text{SHIFT\_3F}(x);$$
$$\text{SHIFT\_2B}(x) \equiv \text{SHIFT\_2F}(x).$$

Множество сдвиговых унарных операций SHIFT_<*i*>ς при ς = {B, F} определено также для $i > 3$ при условии, что для любого количества сдвигов (поворотов) $\omega \in \mathbb{N}$ справедливо равенство (2.2), определяющее значение i как остаток от деления ω на 4:

$$i = \omega \bmod 4. \qquad (2.2)$$

В частности, циклическое вращение вперед или назад на 4 ($i = 4$) условная плоскость логического пространства совершает 4/4 = 1 полный оборот, повторяя его предыдущее состояние. В таком случае, согласно (2.2), $i = 4 \bmod 4 = 0$, и функция циклического сдвига принимает вид SHIFT_0ς при ς = {B, F}.

Такая организация логических сдвиговых операций подчиняется закону *k*-ого сдвига в *k*-значной логике, согласно которому *k* сдвигов вперед/назад равносильны повторению (утверждению) [105]. Ниже приведены примеры унарных тетрафункций сдвиговой группы, возвращающих значение исходного

логического состояния x после поворота логического пространства на $i/4$ оборота при значениях i, приведенных к диапазону значений от 0 до 3:

$$\text{SHIFT\_16B}(x) \equiv \text{SHIFT\_0B}(x);\ \ \text{SHIFT\_25B}(x) \equiv \text{SHIFT\_1B}(x);$$

$$\text{SHIFT\_18F}(x) \equiv \text{SHIFT\_2F}(x);\ \ \text{SHIFT\_31F}(x) \equiv \text{SHIFT\_3F}(x).$$

Унарные функции $\text{MIN\_}\zeta(x)$ и $\text{MAX\_}\zeta(x)$ при $\zeta = \{\text{A, M}\}$ – тетрафункции, приводящие неоднозначное логическое значение x, выраженное состояниями неопределенности или множественности, к определенным (однозначным) значениям логических нуля и единицы.

В качестве минимального возвращаемого логического состояния определено значение логического нуля, а в качестве максимального возвращаемого логического состояния – значение логической единицы.

Таким образом,

$$\text{MIN\_A}(x) = \begin{cases} 0, & if \quad x = \text{A}, \\ x, & if \quad x \neq \text{A}; \end{cases} \qquad \text{MAX\_A}(x) = \begin{cases} 1, & if \quad x = \text{A}, \\ x, & if \quad x \neq \text{A}; \end{cases}$$

$$\text{MIN\_M}(x) = \begin{cases} 0, & if \quad x = \text{M}, \\ x, & if \quad x \neq \text{M}; \end{cases} \qquad \text{MAX\_M}(x) = \begin{cases} 1, & if \quad x = \text{M}, \\ x, & if \quad x \neq \text{M}. \end{cases}$$

2.2.3 Эквивалентность логических операций отрицания и сдвига

Тетрафункция отрицания или инверсии исходного логического состояния x может быть получена альтернативными тетрафункциями как сдвиговой, так и инверсной группы. Из таблицы 2.1 очевидно, что полная или постбинарная инверсия $\overline{x}$ (т. е. функция полного логического отрицания x) тождественно равна циклическому сдвигу вперед или назад на 1/2 оборота двумерного логического пространства:

$$\overline{x} \equiv \text{SHIFT\_2F}(x);\ \overline{x} \equiv \text{SHIFT\_2B}(x).$$

Также постбинарная инверсия $\overline{x}$ может быть достигнута последовательным выполнением одной из двух пар тетрафункций: пара вероятностных инверсий;

пара инверсий с фиксацией нуля и единицы и с фиксацией неопределенности и множественности:

$$\bar{x} \equiv \text{SWAP\_FL}(\text{SWAP\_TR}(x)); \; \bar{x} \equiv \text{SWAP\_0/1}(\text{SWAP\_A/M}(x)).$$

Полученные эквивалентности подтверждают суждение о том, что результатом отрицания в логике является высказывание, в известном смысле противоположное исходному.

При этом последовательность выполнения тетрафункций одной пары не важна, поскольку постбинарная инверсия формируется путем объединения тетрафункций этой пары (Рисунок А.3), а операция объединения ($\cup$) обладает свойством коммутативности.

2.3 Реализация базовых двуместных логических операций

Двуместные функции тетралогики позволяют создать множество полных систем функций представленной логики. Из (2.1) общее число двуместных (бинарных) логических операций тетралогики определено в количестве $N_4^2 = 4^{(4^2)\cdot 1} = 4^{16} > 4$ млрд. (к примеру, в двоичной логике определено всего $2^4 = 16$, а в троичной логике – $3^9 = 19\,683$ двуместных операций).

В частности, для достаточного набора тетрафункций возможна адаптация критерия Поста и для тетралогики, в котором появится возможность сформулировать необходимое и достаточное условие для того, чтобы некоторый набор тетрафункций обладал достаточной выразительностью для представления любой функции тетралогики из полного набора.

Однако следует учесть, что на данный момент в тетралогике, как и в любой k-значной логике не существует полного описания замкнутых классов при $k > 2$. Полное описание системы замкнутых классов для двузначной логики ($k = 2$) было предложено Эмилем Постом в 1940 году [107, с. 9]. Отсутствие замкнутых классов является одним из барьеров, сдерживающих развитие и распространение k-значной логики.

В качестве базовых двуместных тетрафункций можно выделить логические умножение и сложение как операции, которые определенны для непустого множества {0, 1, A, M}.

2.3.1 Логическое умножение в тетралогике

Логическое умножение (конъюнкция, операция «AND») – логическая операция, по своему применению максимально приближенная к союзу «И». В традиционной логике высказывание x AND y истинно тогда и только тогда, когда оба высказывания x, y истинны. Для обозначения логической операции умножения в тетралогике, также как и в двоичной логике, используются все варианты инфиксной записи: $x \wedge y$; $x \,\&\, y$; x AND y; $x \cdot y$, или просто xy. Кроме этого, в качестве одного из способов определения операции конъюнкции может использоваться постфиксная запись $\min(x, y)$ где x, y – константы тетралогики, т. е. $x, y \in \{0, 1, A, M\}$. При $x, y \in \{0, 1\}$, естественно, сохраняется совместимость с операцией конъюнкции двоичной логики.

В контексте выражения $x \wedge y = \min(x, y)$ необходимо определить тетрафункцию, которая в силу наличия неявных, не поддающихся однозначному логическому сравнению, состояний неопределенности и множественности, не столь очевидна, как аналогичная ей двоичная функция.

Основываясь на конъюнкции двоичной логики и учитывая соотношения логических нуля и единицы обеих логик, получаем для тетралогики следующие утверждения:

$$0 \wedge 0 = 0;$$

$$0 \wedge 1 = 0;$$

$$1 \wedge 1 = 1.$$

Учитывая также, что $x \wedge 1 = x$ и $x \wedge 0 = 0$ для любых x, получаем утверждения, в которых пересекаются логические состояния множественности и неопределенности с логическими 0 и 1:

$$\mathrm{A} \wedge 0 = 0;$$

$$M \wedge 0 = 0;$$

$$A \wedge 1 = A;$$

$$M \wedge 1 = M.$$

Учитывая монотонность функции, определяющей операцию конъюнкции, а также очевидное утверждение, что $\min(x, x) = x$, получаем, что

$$A \wedge A = A;$$

$$M \wedge M = M.$$

Учитывая коммутативность операции конъюнкции, представленные выше утверждения определяют 14 сочетаний констант тетралогики из 16-ти возможных. Остается рассмотреть два равнозначных сочетания $A \wedge M$ и $M \wedge A$.

Результат конъюнкции между A и M не столь очевиден, поскольку неочевиден и отчасти парадоксален результат выражения $\min(A, M)$, так как невозможно однозначно утверждать, в каком из состояний A или M больше значений «Ложь» или «Истина». Поэтому целесообразно рассмотреть значения тетралогики 0, A, M и 1 как вершины аппроксимационной решетки AG4 (Approximation Grid) [22, с. 46].

Решетка AG4 является простейшей диаграммой Хассе [108], в которой объединениями и пересечениями являются наименьшие верхние и наибольшие нижние грани соответственно, а операция нестрогого включения ($\subseteq$) повышает логическое значение (Рисунок 2.2).

Логическое значение неопределенности (A) является нижней точкой, поскольку не несет в себе однозначной информации, что в общем случае может трактоваться как полное ее отсутствие. Значение множественности (M) является вершинной, так как дает слишком противоречивую информацию.

Теперь, поскольку $0 \subseteq M$, по монотонности конъюнкции получаем

$$(0 \wedge A) \subseteq (M \wedge A),$$

откуда, в силу свойства $x \wedge 0 = 0$, справедливо выражение

$$0 \subseteq (M \wedge A). \qquad (2.3)$$

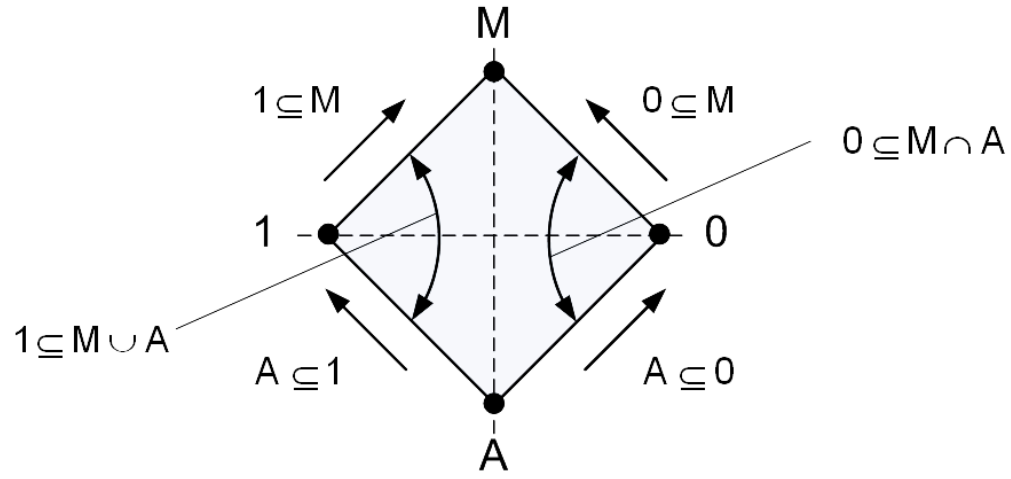


Рисунок 2.2 – Аппроксимационная решетка AG4

Аналогичным образом $A \subseteq 0$ приводит к $(M \wedge A) \subseteq (M \wedge 0)$, откуда

$$(M \wedge A) \subseteq 0. \tag{2.4}$$

Связав соотношения (2.3) и (2.4) операцией конъюнкции и учитывая антисимметричность операции строгого включения, получим равенство

$$\big(0 \subseteq (M \wedge A)\big) \wedge \big((M \wedge A) \subseteq 0\big) \Leftrightarrow M \wedge A = 0. \tag{2.5}$$

Таким образом, благодаря соотношению (2.5), разрешилась неочевидность и парадоксальность выражения min(A, M), результатом которого является логический ноль: min(A, M) = 0. В таблице 2.2 сведены все сочетания конъюнкции тетралогики.

Таблица 2.2 – Таблица истинности конъюнкции в тетралогике ($x \wedge y$)

x \ *y*	0	A	M	1
0	0	0	0	0
A	0	A	0	A
M	0	0	M	M
1	0	A	M	1

2.3.2 Логическое сложение в тетралогике

Логическое сложение (дизъюнкция, операция «OR») – логическая операция, по своему применению максимально приближенная к союзу «ИЛИ» в смысле «или то, или это, или оба сразу». В традиционной логике высказывание x OR y истинно тогда и только тогда, когда хотя бы одно из высказываний x, y истинно.

Для обозначения логической операции сложения в тетралогике, также как и в двоичной логике, используются все варианты инфиксной записи: $x \vee y$; $x \mid y$; $x \parallel y$; $x + y$; x OR y. Также в качестве одного из способов определения операции конъюнкции может использоваться постфиксная запись $\max(x, y)$ где x, y – константы тетралогики. В тетралогике, как и для операции конъюнкции, дизъюнкция сохраняет совместимость с аналогичной операцией двоичной логики при $x, y \in \{0, 1\}$.

Дальнейшее выведение значений дизъюнкции для всех сочетаний тетралогических констант может быть выполнено несколькими возможными путями. Один из них состоит в том, чтобы проделать все этапы работы с основными свойствами дизъюнкции классической логики и разрешить возникнувшие парадоксы с помощью аппроксимационной решетки AG4 (Рисунок 2.7). Однако целесообразно вывести некоторые значения дизъюнкции, связав ее с рассмотренной ранее операцией конъюнкции.

Для связки логических сложения и умножения справедливы следующие эквивалентности:

1) $x \vee y = x$ тогда и только тогда, когда $x \wedge y = y$;

2) $x \vee y = y$ тогда и только тогда, когда $x \wedge y = x$.

Используя приведенные эквивалентности можно определить 14 сочетаний констант тетралогики из 16-ти возможных. Остается рассмотреть два равнозначных сочетания $\mathrm{A} \vee \mathrm{M}$ и $\mathrm{M} \vee \mathrm{A}$. Рассмотрим два варианта:

1. Проведем аналогию с конъюнкцией: если $\min(A, M) = 0$, то $\max(A, M) = 1$, следовательно $A \vee M = M \vee A = 1$.

2. Воспользуемся решеткой AG4 (Рисунок 2.2). Следует отметить, что логическое сложение эквивалентно операции объединения на решетке AG4. Поэтому поскольку $1 \subseteq M$, то по монотонности конъюнкции получаем $(1 \vee A) \subseteq (M \vee A)$, откуда, в силу свойства $x \vee 1 = 1$, справедливо выражение $1 \subseteq (M \vee A)$. Аналогичным образом $A \subseteq 1$ приводит к $(M \vee A) \subseteq (M \vee 1)$, откуда $(M \vee A) \subseteq 1$. Дальнейшие рассуждения вполне очевидны:

$$\big(1 \subseteq (M \vee A)\big) \vee \big((M \vee A) \subseteq 1\big) \Leftrightarrow M \vee A = 1.$$

В таблице 2.3 сведены все сочетания операции дизъюнкции тетралогики.

В монографии [68] получены альтернативные способы определения тетрафункций конъюнкции и дизъюнкции с использованием аксиоматики Цермело-Френкеля (ZF – Zermelo-Fraenkel) [109, с. 157], которые в данной работе представлены в приложении Б.

Таблица 2.3 – Таблица истинности дизъюнкции в тетралогике ($x \vee y$)

x \ y	0	A	M	1
0	0	A	M	1
A	A	A	1	1
M	M	1	M	1
1	1	1	1	1

2.4 Основные свойства функций тетралогкики

Полученные основные функции тетралогики, выполняющие операции отрицания, дизъюнкции и конъюнкции сохранили в себе все важнейшие равносильности алгебры классической логики. Из таблиц 2.2 и 2.3 легко видеть, что конъюнкция и дизъюнкция тетралогики обладают рядом свойств аналогичных

функции булевой алгебры. Например, для $x \in \{0, 1, A, M\}$ справедливо свойство идентичности:

$$x \wedge 1 = x, \tag{2.6}$$

$$x \vee 0 = x. \tag{2.7}$$

Это значит, что согласно выражению (2.6), из тетралогического выражения можно удалить единицу (или часть выражения, равную единице), которая связана с остальным выражением операцией конъюнкции. Выражение (2.7) указывает на то, что из выражения тетралогики можно удалить часть, равную нулю, которая связана с остальным выражением операцией дизъюнкции.

Также сохранились свойства идемпотентности конъюнкции и дизъюнкции (это заметно по результатам основной диагонали для функций в таблицах 2.2 и 2.3):

$$x \wedge x = x, \tag{2.8}$$

$$x \vee x = x. \tag{2.9}$$

Выражение (2.8) можно представить в следующем виде:

$$\bigwedge_{i=0}^{n-1} x = \underbrace{x \wedge x \wedge \ldots \wedge x}_{n \text{ аргументов}} = x^n = x.$$

Выражение (2.9) также может содержать любое количество аргументов:

$$\bigvee_{i=0}^{n-1} x = \underbrace{x \vee x \vee \ldots \vee x}_{n \text{ аргументов}} = n \cdot x = x.$$

Также для логических значений $x, y, z \in \{0, 1, A, M\}$ справедливы свойства коммутативности, ассоциативности и дистрибутивности:

$$x \vee y = y \vee x; \tag{2.10}$$

$$x \wedge y = y \wedge x; \tag{2.11}$$

$$(x \vee y) \vee z = x \vee (y \vee z); \tag{2.12}$$

$$(x \wedge y) \wedge z = x \wedge (y \wedge z); \tag{2.13}$$

$$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z); \tag{2.14}$$

$$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z); \tag{2.15}$$

Равенства (2.10) и (2.12), (2.11) и (2.13) являются свойствами коммутативности и ассоциативности дизъюнкции и конъюнкции соответственно. Ассоциативность дизъюнкции и конъюнкции позволяет опускать скобки в дизъюнкциях и конъюнкциях нескольких переменных, коммутативность – расставлять члены таких дизъюнкций и конъюнкций в любом порядке.

Доказательство свойств коммутативности (2.10) и (2.11) вытекает из симметричности таблиц истинности относительно главной диагонали (см. таблицы 2.2 и 2.3).

При подтверждении свойств ассоциативности имеет место доказательство с использованием метода полной индукции [97, с. 89]. Суть метода заключается в построении таблицы истинности для левой и правой частей доказываемого равенства. Если на всех наборах входных переменных значения левой и правой частей совпадают, то они идентичны и, следовательно, справедливость равенства доказана. В таблице 2.4 показана справедливость ассоциативности дизъюнкции (выделенные столбцы имеют одинаковые логические значения). При этом в таблице используются только исследуемые значения $x, y, z \in \{\text{A}, \text{M}\}$, поскольку использование множества логических значений набора $x, y, z \in \{0, 1\}$ приведет к ожидаемым и уже доказанным в булевой алгебре результатам, тем более, что поллная таблица истинности тетрафункции трех аргументов будет составлять $3^4 = 81$ строку. Аналогичным образом доказывается и свойство ассоциативности конъюнкции.

Равенства (2.14) и (2.15) – это дистрибутивные (распределительные) законы дизъюнкции и конъюнкции, которые позволяют преобразовывать выражения так, чтобы операции в них выполнялись в обратном порядке (например, если в исходном выражении вначале выполнялась дизъюнкция, а потом конъюнкция, то можно получить равносильную формулу, в которой вначале выполняется конъюнкция, а потом дизъюнкция).

Таблица 2.4 – Таблица истинности ассоциативности дизъюнкции

x	y	z	$x \vee y$	$(x \vee y) \vee z$	$y \vee z$	$x \vee (y \vee z)$
A	A	A	A	A	A	A
A	A	M	A	1	1	1
A	M	A	1	1	1	1
A	M	M	1	1	M	1
M	A	A	1	1	A	1
M	A	M	1	1	1	1
M	M	A	M	1	1	1
M	M	M	M	M	M	M

В таблице 2.5 показана справедливость дистрибутивности конъюнкции (2.15). Дистрибутивность дизъюнкции доказывается аналогично, с применением метода полной индукции.

Таблица 2.5 – Таблица истинности дистрибутивности конъюнкции

x	y	z	$y \vee z$	$x \wedge (y \vee z)$	$x \wedge y$	$x \wedge z$	$(x \wedge y) \vee (x \wedge z)$
A	A	A	A	A	A	A	A
A	A	M	1	A	A	0	A
A	M	A	1	A	0	A	A
A	M	M	M	0	0	0	0
M	A	A	A	0	0	0	0
M	A	M	1	M	0	M	M
M	M	A	1	M	M	0	M
M	M	M	M	M	M	M	M

Покажем также справедливость выполнения в тетралогике законов де Моргана [110, 111],

$$\overline{x \vee y} = \bar{x} \wedge \bar{y}; \tag{2.16}$$

$$\overline{x \wedge y} = \bar{x} \vee \bar{y}, \tag{2.17}$$

таблица истинности, которых приведена в таблице 2.6.

Равенства (2.16) и (2.17) позволяют выразить конъюнкцию через дизъюнкцию и отрицание, а дизъюнкцию – через конъюнкцию и отрицание. Эти же соотношения используются для перенесения отрицаний, применяемых к сложным высказываниям, на составляющие их простые [112, с. 35–36].

Поскольку выполнение рассмотренных законов логики позволяет проверять правильность рассуждений и доказательств, а нарушения этих законов приводят к логическим ошибкам и вытекающим из них противоречиям, то таким образом можно утверждать о корректности сформулированных логических операций тетралогики.

Таблица 2.6 – Таблица истинности для теорем де Моргана

x	y	$\bar{x}$	$\bar{y}$	$x \vee y$	$\overline{x \vee y}$	$\bar{x} \wedge \bar{y}$	$x \wedge y$	$\overline{x \wedge y}$	$\bar{x} \vee \bar{y}$
0	0	1	1	0	1	1	0	1	1
0	A	1	M	A	M	M	0	1	1
0	M	1	A	M	A	A	0	1	1
0	1	1	0	1	0	0	0	1	1
A	0	M	1	A	M	M	0	1	1
A	A	M	M	A	M	M	A	M	M
A	M	M	A	1	0	0	0	1	1
A	1	M	0	1	0	0	A	M	M
M	0	A	1	M	A	A	0	1	1
M	A	A	M	1	0	0	0	1	1
M	M	A	A	M	A	A	M	A	A
M	1	A	0	1	0	0	M	A	A
1	0	0	1	1	0	0	0	1	1
1	A	0	M	1	0	0	A	M	M
1	M	0	A	1	0	0	M	A	A
1	1	0	0	1	0	0	1	0	0

Отрицание, конъюнкция и дизъюнкция являются базовыми функциями булевой алгебры. На основании базовых функций можно путем равносильных преобразований получить целый ряд булевых функций двух переменных.

Поэтому тетрафункции отрицания, конъюнкции и дизъюнкции представляют собой функционально полную систему, поскольку, так же как и аналогичные функции классической логики, вписываются в концепцию теоремы Поста – Яблонского [97, с. 191] о признаках полноты системы функций.

Таким образом, в тетралогике с помощью функционально полного базиса тетрафункций, состоящего из отрицания, конъюнкции и дизъюнкции, можно вывести группу тетрафункций, эквивалентных булевым [113, 114]. В таблице 2.7 приведена таблица истинности функций тетралогики с указанием соответствующих преобразований [80, 115]:

1) $x \oplus y = (\overline{x} \wedge y) \vee (x \wedge \overline{y})$ – сумма по модулю 2 (исключающее «ИЛИ);

2) $x \downarrow y = \overline{x \vee y}$ – отрицание дизъюнкции (функция «ИЛИ-НЕ»);

3) $x \leftrightarrow y = (\overline{x} \vee y) \wedge (x \vee \overline{y})$ – эквиваленция (или эквивалентность);

4) $x \rightarrow y = \overline{x} \vee y$, $x \leftarrow y = x \vee \overline{y}$ – импликация (следование);

5) $x \mid y = \overline{x \wedge y}$ – отрицание конъюнкции (функция «И-НЕ»).

Таблица 2.7 – Таблица истинности тетрафункций двух переменных $Tr^2(x, y)$

x	0	0	0	0	A	A	A	A	M	M	M	M	1	1	1	1
y	0	A	M	1	0	A	M	1	0	A	M	1	0	A	M	1
$x \oplus y$	0	A	M	1	A	0	1	M	M	1	0	A	1	M	A	0
$x \downarrow y$	1	M	A	0	M	M	0	0	A	0	A	0	0	0	0	0
$x \mid y$	1	1	1	1	1	M	1	M	1	1	A	A	1	M	A	0
$x \rightarrow y$	1	1	1	1	M	1	M	1	A	A	1	1	0	A	M	1
$x \leftarrow y$	1	M	A	0	1	1	A	A	1	M	1	M	1	1	1	1
$x \leftrightarrow y$	1	M	A	0	M	1	0	A	A	0	1	M	0	A	M	1

2.5 Выводы по второй главе

Тетралогика представляет собой существенно расширенную логическую систему по сравнению с традиционной бинарной логикой. Это связано, прежде всего, с тем, что в тетралогике имеются функции, не выразимые в двузначной логике.

Рассмотренные основные операции тетралогики выступают в качестве математического аппарата постбинарной логики, что позволяет заключить следующее:

1. Операция дизъюнкции тетралогики сохранила правила дизъюнкции классической алгебры логики: результат равен 0, если все операнды равны 0; результат равен 1, если хотя бы один из операндов равен 1.
2. Операция конъюнкции тетралогики также сохранила правила конъюнкции классической алгебры логики: результат равен 1, если все операнды равны 1; результат равен 0, если хотя бы один из операндов равен 0.
3. Дизъюнкция и конъюнкция могут быть как бинарными тетрафукцциями (иметь два аргумента), так и тернарными (иметь три аргумента) или *n*-арными (иметь *n* аргументов).
4. Унарные операции тетралогики, объединенные в инверсную и сдвиговую группы, и двуместные поразрядные операции тетралогики реализованы с соблюдением основных законов логики:
   - двойного отрицания для унарных операций инверсной группы;
   - *n*-го сдвига *n*-значных логических системах для унарных операций сдвиговой группы;
   - коммутативности, ассоциативности и дистрибутивности для бинарных операций.
5. Рассмотренные унарные и бинарные функции алгебры тетралогики представляют собой теоретическую основу, на базе которой возможна

аппаратная реализация базовых логических элементов с унарным выходом.

Следует также отметить, что высказывания в тетралогике могут быть истинными, ложными, содержащими или истину или ложь, содержащими и истину и ложь одновременно. Это существенно сближает высказывания тетралогики с философской сущностью бытия и расширяет применение как соответствующей логики, так и расширенного кодо-логического базиса в целом. Поэтому с целью обеспечения более реалистичного высокопроизводительного компьютерного моделирования сложных систем становится также возможным создание вычислительных моделей, способных представить в вычисляемом виде большинство противоречий окружающего мира.

ГЛАВА 3
ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ ТЕТРАКОДИРОВАНИЯ

3.1 Основные принципы тетракодирования

Состояния тетралогики могут кодироваться тетракодом, представленным в виде значений $C_4 = \{0, A, M, 1\}$ и используемым по аналогии с бинарным кодом для поразрядного представления количественных значений. При этом в отличие от двоичной логики, где n-разрядное пространство определено 2^n двоичными значениями, в тетралогике количество увеличивается до $2^{2n} = 4^n$ значений тетракодов.

По аналогии с понятием «бит» (англ. *binary digit* – один разряд в двоичной системе счисления) для поразрядного представления тетракода используется понятие «тетрит» (англ. *tetrit*), являющееся соответствующей трансформацией корневой основы «тетра» (англ. *tetra*), связанной со значением слова «четыре».

Следует отметить, что параллельно понятие «tetrit» было предложено использовать как средство для упрощения и разъяснения семантики обработки исключительных ситуаций при интервальных вычислениях [116], что в дальнейшем можно рассматривать как альтернативный (более узкий) вариант использования данного термина. Также, согласно [117, с. 16], тернарная энтропия H_4 источника S с исходным алфавитом $\{a_1, a_2, a_3, a_4\}$ равна одному тетриту:

$$H_4(S) = -\sum_{i=1}^{4} p_i \log_4 p_i = -\sum_{i=1}^{4} \frac{1}{4} \log_4 \frac{1}{4} = -\log_4 \frac{1}{4} = 1 \text{ тетрит}, \qquad (3.1)$$

где p_i является вероятностью a_i: $p_i = p(a_i)$.

Четверичная система кодирования количественной информации (тетракод) может быть определена различными комбинациями из четырех разрядов (тетритов), кодирующими состояния двумерного логического пространства в различных сочетаниях [28, с. 289], [30]. Однако тетракод $C_4 = \{0, A, M, 1\}$ получил

более широкое применение в ряде исследований [118, 119, 120, 121]. Поэтому под тетракодом будет в дальнейшем пониматься способ представления данных в одном разряде в виде комбинации четырех знаков, обозначаемых цифрами 0, 1 и буквами «А», «М». Тетракод является позиционным кодом, поэтому число комбинаций (кодов) *k*-разрядного тетракода равно числу размещений с повторениями:

$$\tilde{A}_4^k = 4^k, \tag{3.2}$$

где k – число разрядов тетракода, т. е. количество тетритов.

Например, используя два тетрита можно закодировать 16 различных комбинаций: 00 0A 0M 01 A0 AA AM A1 M0 MA MM M1 10 1A 1M 11; три разряда – 64: 000 00A 00M 001 … 110 11A 11M 111, и т. д.

Таким образом, для постбинарных вычислений и постбинарного кодирования примем следующие определения:

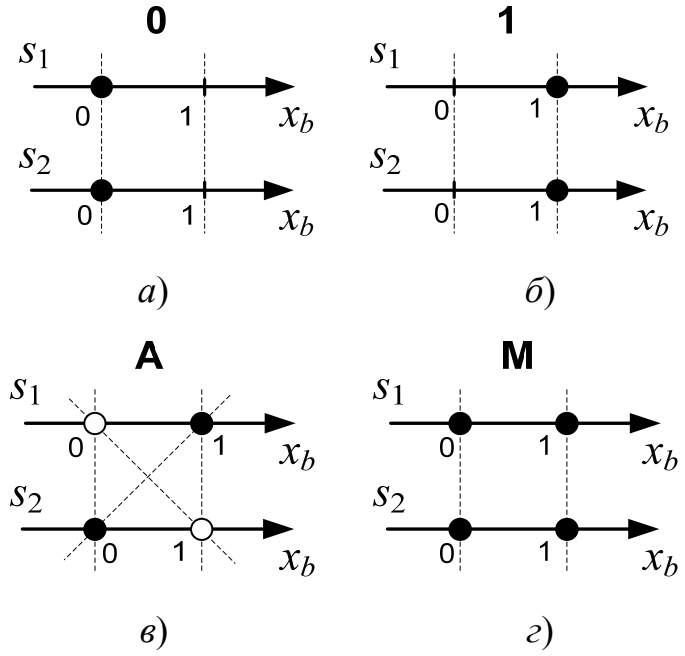
Определение 1. Постбинарные вычисления – это компьютерные вычисления, в которых в качестве входных данных и полученных результатов выступают постбинарные коды, отображающие числовые данные.

Определение 2. Постбинарное кодирование – кодирование количественной информации с помощью постбинарных кодов. Далее в работе под постбинарными кодами будем понимать тетракоды.

При рассмотрении тетракода в качестве носителя количественной информации, представленной в двоичном коде, можно высказать следующие утверждения:

- тетриты 0 и 1 приводятся к битам 0 и 1 соответственно, так как они кодируют аналогичные значения;
- тетрит М приводится к битам 0 и 1 одновременно, поэтому представляется двумя точками на числовой оси;
- тетрит А приводится к битам 0 или 1 (в момент приведения его значение неизвестно), поэтому представляется одной из двух возможных точек на числовой оси.

На рисунке 3.1 показано, как тетрит t приводится к биту b на бинарной оси x_b по событиям s_1 и s_2 – выборкам двоичных значений из тетракода в разные моменты времени. При этом тетриты 0 и 1 можно назвать однозначно представимыми на числовой оси, поскольку их значения сводятся к аналогичным двоичным значениям 0 и 1 в любой момент времени (Рисунок 3.1 *а*, *б*), что подтверждено идентичностью состояний «Истина» и «Ложь» тетралогики и классической бинарной логики.



а) *б*) *в*) *г*)

Рисунок 3.1 – Позиционирование тетритов на числовой оси x_b (*а* – приведение однозначно представимого тетрита 0 к двоичному нулю; *б* – приведение однозначно представимого тетрита 1 к двоичной единице; *в* – приведение неоднозначно представимого тетрита А к двоичным 0 или 1; *г* – приведение неоднозначно представимого тетрита М к двоичным 0 и 1)

Тетриты А и М являются неоднозначными, поскольку не могут быть приведены к определенному двоичному значению. Тетрит А занимает одно случайное двоичное значение 0 или 1 в каждом событии, т. е. он всегда позиционируется одной из двух возможных точек на числовой оси. На рисунке 3.1 *в* тетрит А при наступлении события s_1 был случайным образом приведен к двоичной 1 (с такой же вероятностью он мог бы занять позицию

двоичного 0), а при наступлении события s_2 принял значение двоичного нуля (с такой же вероятностью он мог бы сохранить предыдущее значение от события s_1). Данный аспект представления тетрита A позволяет эффективно кодировать состояния тетралогики, обладающие свойствами равновероятности, а при совокупности множества выборок – свойствами случайности. Тетрит M приводится к паре значений 0 и 1 во всех событиях s_i, т. е. он всегда позиционируется двумя точками на числовой оси (Рисунок 3.1 *г*). Поэтому два тетрита M позиционируются четырьмя точками на числовой оси; три тетрита M – $2^3 = 8$ точками; n тетритов M – 2^n точками на числовой оси.

Графическая интерпретация приведения 8-разрядного тетракода к одному или совокупности двоичных кодов размерности [7:0], по аналогии с полями постбинарного клеточного автомата [121, 122, 123], приведена в приложении B.

На рисунке 3.2 представлены результаты программного приведения тетракода к двоичному коду с последующим преобразованием к десятичному целому числу без знака [124]. На основании полученных результатов можно сформулировать следующие заключения, определяющие *основные принципы тетракдирования*:

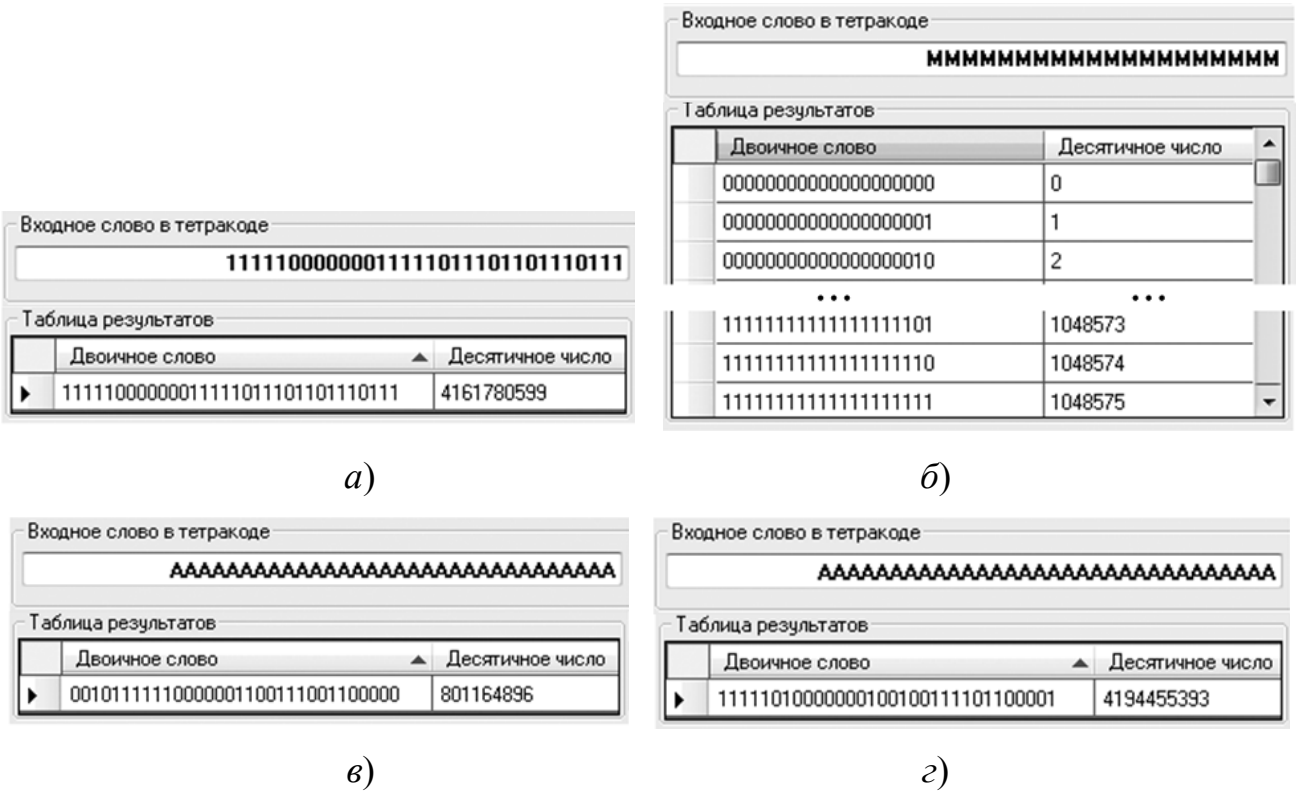
1. При приведении n-разраядного тетракода, состоящего только из значений 0 и/или 1, к двоичному коду, получится один код, при котором каждый i-й тетрит ($i = \overline{0, n-1}$) будет заменен эквивалентным ему битом 0 и/или 1 (Рисунок 3.2 *а*).

2. При приведении n-разраядного тетракода, состоящего только из значений M, к двоичному коду, получится совокупность кодов, при котором каждый i-й тетрит ($i = \overline{0, n-1}$) будет поочередно заменен битами 0 и 1. При этом совокупность кодов представляет собой размещения с повторениями из $m = 2$ элементов (это 0 и 1 к которым приводится тетрит M) по n и представляет собой упорядоченные n-элементные выборки, в которых элементы могут повторяться. Количество таких размещений определяется следующим равенством:

$$\tilde{A}_m^n = \tilde{A}_2^n = 2^n, \tag{3.3}$$

где n – количество тетритов в одном тетракоде, равных M.

Например, 20-разрядный тетракод, состоящий только из тетритов M, приводится к совокупности, состоящей из 2^{20} упорядоченных двоичных кодов в диапазоне от 00000h до FFFFFh, что соответствует диапазону $0 \div (2^{20} - 1)$ десятичных значений (Рисунок 3.2 *б*).



а) *б*)

в) *г*)

Рисунок 3.2 – Результаты приведения тетракода к двоичному коду с использованием программы «Перевод тетракода в 2/10 форматы чисел» (*а* – приведение *n*-разраядного тетракода, состоящего только из значений 0 и/или 1; *б* – приведение *n*-разраядного тетракода, состоящего только из значений M; *в, г* – приведение *n*-разраядного тетракода, состоящего только из значений A, к двоичному коду, при котором каждый тетрит заменен битами 0 или 1 случайным образом)

3. При приведении *n*-разраядного тетракода, состоящего только из значений A, к двоичному коду, получится один код, при котором каждый *i*-й тетрит $(i = \overline{0, n-1})$ будет заменен битами 0 или 1 случайным образом. При этом в

результате повторного приведения такого тетракода (фактически, наступление нового события s_i) может быть получен другой двоичный код с другим значением.

Например, на рисунке 3.2, *в-г* продемонстрировано, как из 32-разрядного тетракода AAA…AA каждый раз получается один 32-разрядный бинарный код, имеющий разные двоичные (а, следовательно, и десятичные) значения.

Полученные заключения в своей совокупности справедливы и для тетракодов, состоящих из любых сочетаний значений $\{0, A, M, 1\}$. Также эти заключения справедливы и для тетракодов, которые после приведения их к двоичным кодам представляют стандартные форматы чисел с плавающей запятой. На рисунке 3.3 представлены результаты приведения тетракода к двоичному коду [124] – формату числа с плавающей запятой одинарной точности с последующим преобразованием к вещественному числу в десятичной системе счисления.

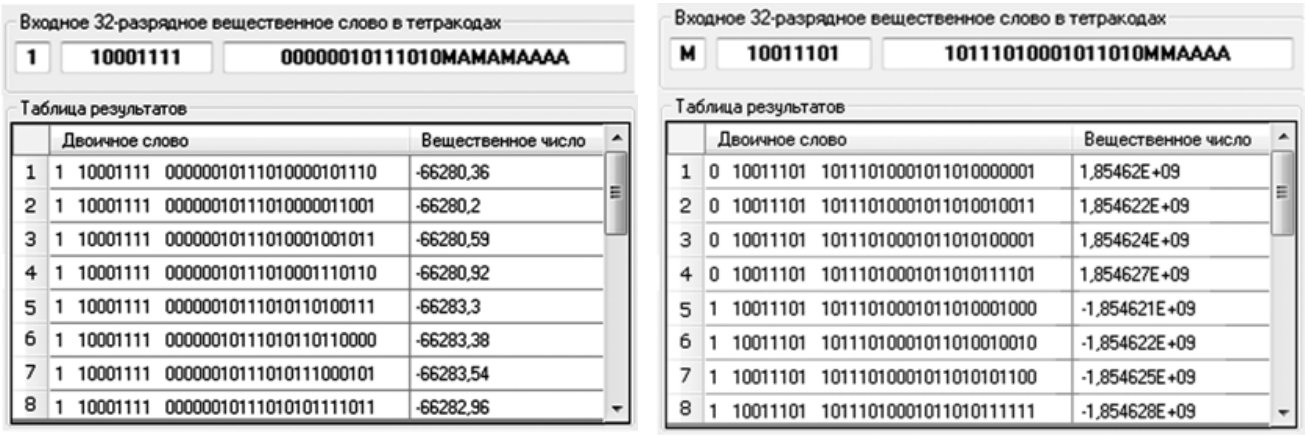


Рисунок 3.3 – Результаты декодирования тетракодов: получение набора двоичных чисел в формате плавающей запятой одинарной точности с использованием программы «Перевод тетракода в 2/10 форматы чисел»

Таким образом, учитывая особенности перевода данных из одной кодовой системы в другую, можно сформулировать следующие уточняющие определения:

Определение 3. Кодирование тетракода – процесс приведения одного или нескольких двоичных кодов к одному тетракоду.

Определение 4. Декодирование тетракода – процесс, обратный кодированию тетракода, т. е. получение одного или нескольких двоичных кодов из одного тетракода.

Определение 5. Запись тетракода в *k*-ричной системе счисления – это поразрядная замена тетракода одним или несколькими значениями *k*-ричной системы.

3.2 Формирование принципов записи тетракода с использованием двоичных кодов

Реализация постбинарных вычислений, в частности возможность работы с тетракодами на современных ЭВМ (которые базируются на бинарной логике и арифметике), невозможна без использования двоичного кодирования разрядов тетракода.

При двоичном кодировании *n* битами можно закодировать $k = 2^n$ возможных состояний (значений). Поэтому для кодирования четырех значений тетракода при $k = 4$ необходимо $n = \log_2 k = \log_2 4 = 2$ бита.

В работах [125, 126] использовалось следующее пары бит для кодирования одного тетрита: для тетрита А – биты 00; для тетрита М – биты 11; для тетрита 0 – 01; для тетрита 1 – 10. Подобное сопоставление пары бит определенному тетриту определяется при сопоставлении состояний двумерного логического пространства и множества точек двумерного евклидового пространства, т. е. точек гиперкуба размерности 2 (2-куба), битовые значения каждой точки которого фактически соответствуют координатам единичного квадрата (Рисунок 3.4).

Поскольку два бита кодируют один разряд тетракода, то каждая двоичная тетрада содержит в себе значения двух тетритов. Следовательно, каждую пару тетритов можно также представлять одним шестнадцатиразрядным значением. В таблице 3.1. приведены соответствия между шестнадцатиразрядным числом, двоичной тетрадой и парой тетритов.

На основании вышесказанного можно утверждать следующее: *m*-разрядный тетракод может приводиться к одному или нескольким двоичным *m*-битным числам, но имеет единственную 2*m*-битную двоичную запись.

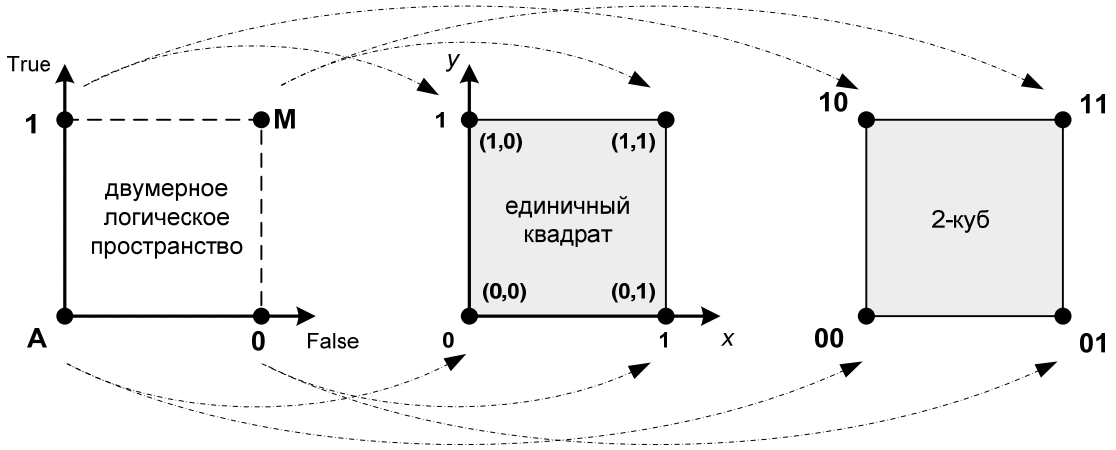


Рисунок 3.4 – Сопоставление состояний двумерного логического пространства координатам единичного квадрата и вершинам 2-куба

Например, 8-разрядный тетракод 1011**M**AAA, согласно основам тетракодирования, приводится к набору из двух 8-битных двоичных чисел, например {1011**0**101, 1011**1**010}, а в памяти компьютера может храниться только в виде 16-битного двоичного кода: 1001101011000000.

Часто в текстах для обозначения двоичных и шестнадцатеричных кодов используют прописные буквы «b» и «h» соответственно. Поэтому также целесообразно использовать букву «t» (первая буква англ. *tetra*), чтобы обозначать тетракод в текстовой документации, в том числе и в данной работе.

Например, если 10101001b – бинарный код, A9h – hex-код, 10101001t – тетракод, то справедлива следующая тождественность:

$$10101001t \equiv 10101001b \equiv A9h.$$

Однако тетракод 10101001t может быть представлен в двоичном и шестнадцатеричном коде следующим образом:

$$1001\ 1001\ 1001\ 0110b = 9996h.$$

В последнем случае отсутствует идентичность со значением, кодируемым тетракодом:

$$10101001t \not\equiv (1001\ 1001\ 1001\ 0110b = 9996h),$$

поскольку $10101001t = 10101001b$ и $10101001t \neq 1001100110010110b$.

Таблица 3.1 – Таблица кодирования разрядов тетракода (Tetra) с использованием шестнадцатеричных (Hex) и двоичных (Bin) значений

Hex	*Bin*	*Tetra*	*Hex*	*Bin*	*Tetra*
0	0000	AA	8	1000	1A
1	0001	A0	9	1001	10
2	0010	A1	A	1010	11
3	0011	AM	B	1011	1M
4	0100	0A	C	1100	MA
5	0101	00	D	1101	M0
6	0110	01	E	1110	M1
7	0111	0M	F	1111	MM

3.3 Формирование способов постбинарного кодирования десятичных чисел

Частым случаем ошибок представления, имеющим инженерный характер, являются ошибки перевода из одной системы счисления в другую. Многие десятичные дроби не имеют точного конечного представления среди двоичных чисел, с которыми оперируют современные компьютерные системы. Но при введении подобных чисел в ЭВМ они заменяются некоторым конечным рядом по степеням двойки, что, как следствие, вносит ошибку в машинное представление числа.

Простейшей и наиболее распространенной ситуацией описания неизвестной точной величины является задание множества ее возможных значений [41].

Например, величина *a*, представляющая неопределенность значений на числовой прямой между 1 и 5, может быть задана $1 \le a \le 5$ или, в обобщенном виде, как $a \in \mathbb{R}$.

Потребность такого представления неопределенности возникает во множестве самых разнообразных ситуаций, в частности при представлении в машинных кодах бесконечных десятичных дробей. Например, число π – бесконечную десятичную дробь можно представить как некоторое приближение вещественного числа, имеющее конечное число десятичных или двоичных знаков, например $\pi \approx 3{,}14159$. Однако при оперировании таким приближением числа π уже в начале вычислений допускается неизбежная ошибка, поскольку неизвестна информация о том, каким именно приближением, с недостатком или с избытком, является указываемое значение. Очевидно, что при более корректном представлении данных нужно явным образом указывать границу этой ошибки, например, путем уточнения того, что ошибка представления не превосходит половины единицы последнего разряда. Существует более предпочтительный способ указания ошибки представления, который состоит в том, чтобы предоставить наиболее узкие точечно-представимые границы (нижнюю и верхнюю) для задания какой-либо неоднозначной величины. Так, для числа π:

$$\pi \in [3.14159, 3.14160].$$

Такой подход называется «интервальным» и применяется к решению различных типов задач, при котором использование машинной интервальной арифметики уменьшает (или вовсе устраняет) проблемы соотнесения вычисленного приближенного результата с истинным (абстрактным) решением. При таком подходе интервал, представляемый парой вещественных чисел, является простейшим видом конечнопредставимого множества, локализующего простейший абстрактный объект – вещественное число. Основная идея интервального подхода состоит в том, что вещественное число представляется в памяти вычислительной машины не одним, а двумя машинными числами – нижней и верхней оценкой, образующими интервальное число [127, 128, 129].

Таким образом, в рамках интервального подхода исходные данные и промежуточные результаты представляются граничными значениями, над которыми и проводятся все операции. При этом сами операции определяются таким образом, чтобы результат соответствующей операции обязательно располагался внутри вычисляемых границ [120, 130].

В интервальных вычислениях переход к тетракодам, прежде всего, обусловлен возможностью кодирования совокупности значений в одном поле данных. Кроме того, такая система кодирования количественной информации, обладает по сравнению с традиционными системами набором качественных преимуществ [28, с. 293].

В [127] рассмотрен процесс перехода от десятичного интервала к тетракоду на примере вычисления иррационального числа π с помощью формулы Бэйли-Боруэйна-Плаффа [131]:

$$\pi = f(n) = \sum_{k=0}^{n} \frac{1}{16^k} \cdot \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right). \qquad (3.4)$$

При этом исходный десятичный интервал $\boldsymbol{x} = [x_1, x_2]$ ($\boldsymbol{x} \in \mathbb{IR}$, $x_1, x_2 \in \mathbb{R}$) составим из чисел с 8-ю значащими цифрами, полученными на первом и сотом шагах вычислений выражения (3.4):

$$x_1 = f(1) = 3{,}1414225; \quad x_2 = f(100) = 3{,}1415927.$$

Поскольку полученные значения различаются, начиная с одной десятитысячной, то при использовании этого интервала в процессе вычислений, позволяющих проводить автоматический учет всех видов погрешностей, гарантируется точность четырех первых цифр самого результата, т. е. в данном случае числа π.

Двоичные отображения чисел x_1 и x_2, представленные для наглядности 32-разрядными двоичными дробями без округления, позволяют выразить интервал $\boldsymbol{x}$ = [3,1414225; 3,1415927] одним тетракодом T:

$$x_1 = 3{,}1414225_{10} = 11.00100100001\underline{1}0100010000111110101\text{b},$$
$$x_2 = 3{,}1415927_{10} = 11.00100100001\underline{1}1110110101101001\text{1b},$$

откуда $T = 11.001001000011\text{MMMMMAAAAAAAAAAAAA}$.

Позиция первого (старшего) значения М определена первым несовпадением битовых значений в соответствующих разрядах (в двоичных значениях x_1 и x_2 эти разряды подчеркнуты). Эта позиция определяет точность исходных десятичных чисел:

$$m = \left\lfloor \lg 2^n \right\rfloor, \tag{3.5}$$

где m – количество одинаковых, начиная со старшего разряда, десятичных значащих цифр в исходной паре чисел; n – количество бит, при которых будет соблюдаться требуемая точность.

Так, для данного примера, старший разряд М тетракода приходится на 15-й бит, начиная со старшего разряда, чисел x_1 и x_2. Следовательно, $m = \left\lfloor \lg 2^{15} \right\rfloor = 4$. Действительно, границы интервала [3,1414225, 3,1415927] имеют 4 общие значащие цифры, и различаются, начиная с пятой.

Количество разрядов «многозначности» М обусловлено обеспечением необходимой плотности «отсчетов» внутри границ интервала, обеспечивающее приближение минимум одного из множества вычисляемых значений к исходному числу с заданной точностью.

Необходимым и достаточным условием обеспечения плотности «отсчетов» внутри границ интервала $\boldsymbol{x}$ является использование середины (центра) интервала $\operatorname{mid}\boldsymbol{x}$ – значения, равноудаленного от граничных значений интервала и определяемого равенством (1.8). На рисунке 3.5 показан процесс формирования тетракода T с использованием двоичных значений x_1, x_2 и $\operatorname{mid}\boldsymbol{x}$. Позиция старшего М определяется равенством (3.5) или первым несовпадением значений разрядов одного веса. Количество разрядов М при этом определяется исходя из количества несовпадающих по значению разрядов. Последующее совпадение значений разрядов одного веса означает начало группы незначащих разрядов, которые в тетракоде представлены значением А.

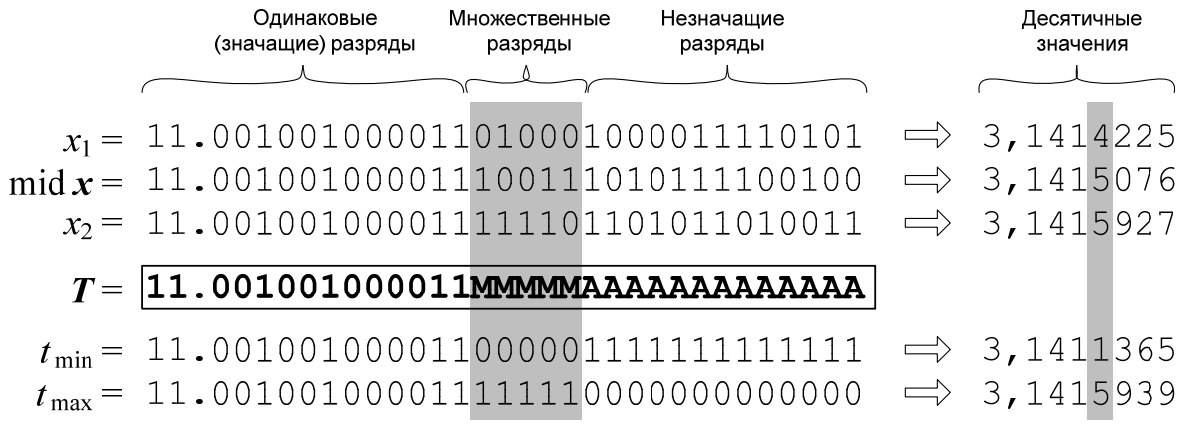


Рисунок 3.5 – Кодирование интервала $\boldsymbol{x}$ тетракодом T с последующим получением границ $t_{\min}$, $t_{\max}$ результирующего интервала

Декодирование тетракода T к результирующему интервалу $\boldsymbol{t} = [t_{\min}, t_{\max}]$ достигается рассмотренными в разделе 2 поразрядными функциями абсолютной минимизации и максимизации M для получения левой и правой границ интервала соответственно. С позиции неопределенности A для границ интервала $\boldsymbol{t}$ рассматриваются такие значения, при которых интервал будет иметь минимальную ширину. Очевидно, что для нижней границы обеспечением минимальной ширины интервала будет тот случай, когда все разряды A примут значение 1, а для верхней границы – когда все разряды A примут значение 0 (Рисунок 3.5).

Таким образом, двоичные значения границ результирующего интервала $\boldsymbol{t}$:

$t_{\min}$ = MIN_M(MAX_A(T)) = 11.00100100001100000111111111111b;

$t_{\max}$ = MAX_M(MIN_A(T)) = 11.00100100001111111000000000000b.

Работа функций минимизации и максимизации в подобном сочетании будет детально рассмотрена в следующем пункте, посвященном декодированию тетракодов.

Полученные десятичные значения

$$t_{\min} = 3{,}1411365;\ \ t_{\max} = 3{,}1415939$$

позволяют утверждать, что тетракод T кодирует границы интервала $\boldsymbol{t}$, который в совою очередь гарантированно содержит все значения исходного интервала $\boldsymbol{x}$. Действительно, полученные результаты соответствуют неравенствам $t_{\min} < x_1$ и

$t_{max} > x_2$. Следовательно, приведенные из T значения t_{min} и t_{max} фиксируют все множество значений функции $f(n)$ выражения (3.4). В этом легко убедиться, поскольку истинное значение $\pi = 3{,}1415926536\ldots$

Полученные результаты являются доказательством следующей леммы:

Лемма 1. При кодировании исходного десятичного интервала $\boldsymbol{x}$ тетракодом T и последующим приведением этого тетракода к результирующему десятичному интервалу $\boldsymbol{t}$ справедливо следующее соотношение

$$\boldsymbol{t} \supseteq \boldsymbol{x}. \tag{3.6}$$

Выражение (3.6) леммы 1 лежит в основе постбинарного кодирования десятичных чисел. При формировании тетракода можно руководствоваться следующими высказываниями:

1. В качестве исходного значения может выступать не только интервальное, а также и традиционное (точечное) значение, поскольку его можно представить в виде вырожденного интервала. В полученном тетракоде будут отсутствовать разряды M и A, так как границы вырожденного интервала равны. Поэтому в условии (3.6) исходный и результирующий интервалы связаны операцией нестрогого включения.
2. Тетракод, ввиду своеобразной замены тетрита A битами 1 или 0 в зависимости от минимальной или максимальной границы интервала, позволяет получить результирующий интервал минимальной ширины. Поскольку тетракодирование подразумевает при переводе тетракода в двоичный код заполнение тетрита A случайными 0 или 1, то полученный при таком подходе интервал всегда будет содержать все значения результирующего интервала, что не противоречит условию (3.6).

На рисунке 3.6 приведен пример формирования тетракода T из исходного интервала $\boldsymbol{y} = [222{,}123;\ 222{,}124]$ с последующим получением результирующих интервалов $\boldsymbol{t}$ и $\boldsymbol{t}'$. Причем $\boldsymbol{t}'$ получен согласно принципам тетракодирования. Поэтому полученная из приведенных на рисунке 3.6 неравенств зависимость

$$\boldsymbol{t}' \supset \boldsymbol{t} \supset \boldsymbol{y}. \tag{3.7}$$

подтверждает приведенное выше высказывание № 2 и не противоречит (3.7).

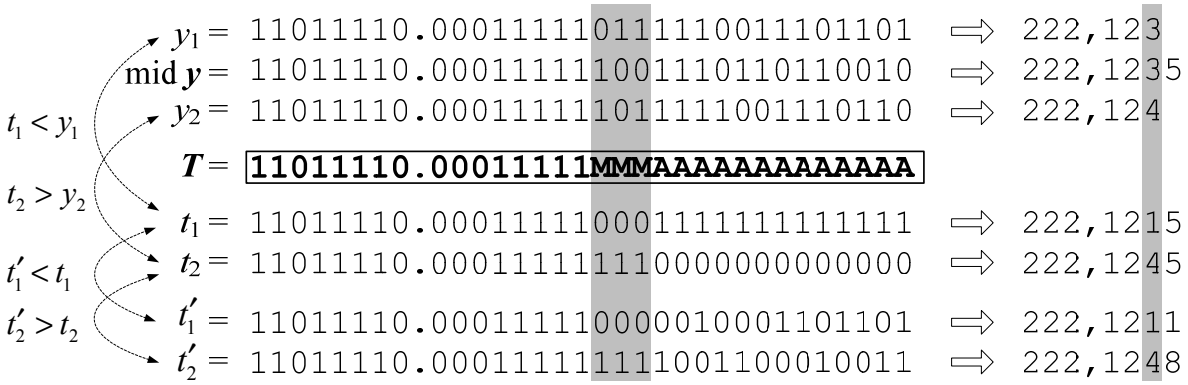


Рисунок 3.6 – Кодирование интервала $\boldsymbol{y}$ тетракодом T с последующим декодированием к интервалам $\boldsymbol{t}$ и $\boldsymbol{t'}$

Представление границ интервалов тетракодом дает возможность гибкого задания практически всех наборов числовых значений, включенных в этот интервал. Несмотря на то, что количество бит, требуемых для «понимания» тетракодов современными ЭВМ, удваивается по сравнению с двоичным кодом, повышение степени информативности получаемых за счет этого тетракодов вполне оправдывает увеличение затрат на кодирование [115].

3.4 Разработка методов приведения тетракодов к интервальным значениям

Кодирование границ невырожденного вещественного интервала тетракодом возможно лишь в том случае, если в последнем присутствуют разряды М, являющиеся определяющим фактором выявления интервальных границ. В таком тетракоде также допускается наличие тетритов А, которые являются уточняющим фактором при выявлении интервальных границ и занимают младшие разряды тетракода, уменьшая тем самым погрешность представления закодированных в тетракоде значений.

Подобная структура тетракода с позиции расположения тетритов М и А наделяет его так называемой *нормированностью*, которая определена совокупностью следующих критериев:

1. Тетриты 0 и 1 являются основной информационной составляющей и фактически определяют позицию числа на числовой оси.
2. Обязательное наличие «группы М» – одного или нескольких подряд идущих тетритов М. Эта группа необходима для формирования границ интервала: она неразрывна и, если отсутствуют разряды А, располагается в младших разрядах тетракода.
3. Необязательное наличие «группы А» – одного или нескольких подряд идущих тетритов А. Группа А носит уточняющий характер для представления границ интервала: она неразрывна и занимает более младшие разряды тетракода, чем группа М.
4. Между группами М и А не допускается появления однозначно представимых тетритов 0 и 1.

Нарушение хотя бы одного из данных критериев говорит о ненормированности тетракода (Рисунок 3.7 *б*). Тетракод, состоящий только из значений 0 и 1 называется вырожденным и приводится, соответственно, к вырожденному интервалу, т. е. к одному значению (Рисунок 3.7 *в*). И только в случае соблюдения нормированности тетракода (Рисунок 3.7 *а*) справедливо его приведение к границам интервала.

Процесс приведения нормированного тетракода к значениям границ интервального числа (декодирование тетракода) является задачей, обратной к рассмотренной в п. 3.3 задаче по приведению пары десятичных чисел к тетракоду (постбинарное кодирование). При декодировании тетракода предполагается, что определение диапазонов возможного изменения границ и нахождение ширины интервала, представленного нормированным тетракодом, является достаточным условием для оценки эффективности и точности позиционирования интервала на числовой оси.

Рассмотрим нормированный тетракод, содержащий только группу М в младших разрядах. При сведении такого тетракода к множеству двоичных значений границы интервала определены крайними позициями этого множества: левая граница – при всех значениях М, приведенных к двоичным нулям; правая

граница – при всех значениях M, приведенных к двоичным единицам. При этом ширина wid полученного интервала $\boldsymbol{x}$ может быть вычислена по формуле (1.6), но поскольку однозначно представимые в исходном тетракоде тетриты 0 и 1 самоуничтожатся операцией вычитания, оценку ширины закодированного интервала можно получить, не используя численные значения x_1 и x_2 (Рисунок 3.8 *а*).

Рисунок 3.7 – Примеры нормированных (*а*), ненормированных (*б*) и вырожденных (*в*) тетракодов

Так как значение тетракода сводится к отображению двоичных чисел, а двоичная система счисления является позиционной, то ширина полученного интервала зависит, прежде всего, от позиции группы M в тетракоде, а точнее – от позиции старшего тетрита поля M. Поэтому

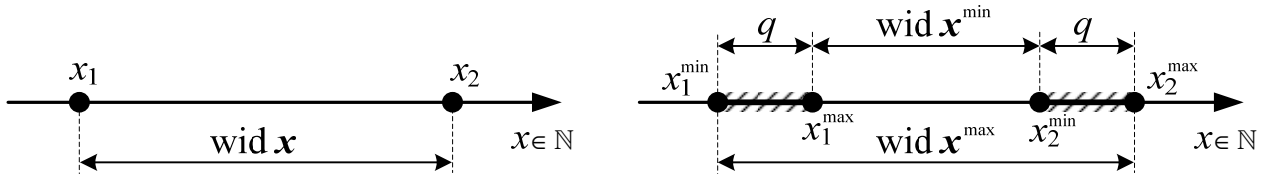
$$\mathrm{wid}\,\boldsymbol{x} = 2^{k+1} - 1, \tag{3.8}$$

где k – позиция старшего тетрита равного M в n-разрядном тетракоде: $n-1 \le k \le 0$.

Нормированный тетракод, содержащий поля M и A при декодировании предполагает определение расстояния q между значениями минимально и максимально возможных границ $x_1^{\min}$, $x_2^{\min}$ и $x_1^{\max}$, $x_2^{\max}$ для интервалов

$\boldsymbol{x}^{\min} = [x_1^{\max}, x_2^{\min}]$ и $\boldsymbol{x}^{\max} = [x_1^{\min}, x_2^{\max}]$, имеющих ширину $\operatorname{wid}\boldsymbol{x}^{\min}$ и $\operatorname{wid}\boldsymbol{x}^{\max}$ соответственно (Рисунок 3.8, *б*):

$$q = x_1^{\max} - x_1^{\min} = x_2^{\max} - x_2^{\min}. \tag{3.9}$$



а) *б)*

Рисунок 3.8 – Формирование целочисленных интервальных границ при декодировании нормированного тетракода содержащего только группу М (*а*) и группы М и А (*б*)

При декодировании нормированного тетракода с полями М и А к интервальным границам, расстояние $\operatorname{wid}\boldsymbol{x}^{\max}$ между крайними точками $x_1^{\min}$ и $x_2^{\max}$ определяется аналогично (3.9), поскольку поле А приводится к таким же двоичным значениям как и для поля М (обнуляются для левой и устанавливаются в единицу для правой границ интервала): $\operatorname{wid}\boldsymbol{x}^{\max} = \operatorname{wid}\boldsymbol{x}$. Расстояние $\operatorname{wid}\boldsymbol{x}^{\min}$ между точками $x_1^{\max}$ и $x_2^{\min}$ связано с $\operatorname{wid}\boldsymbol{x}^{\max}$ соотношением

$$\operatorname{wid}\boldsymbol{x}^{\min} = \operatorname{wid}\boldsymbol{x}^{\max} - 2q, \tag{3.10}$$

откуда

$$q = \frac{1}{2} \cdot (\operatorname{wid}\boldsymbol{x}^{\max} - \operatorname{wid}\boldsymbol{x}^{\min}). \tag{3.11}$$

Равенство (3.11) определяет q как расстояние между двумя интервалами шириной $\operatorname{wid}\boldsymbol{x}^{\min}$ и $\operatorname{wid}\boldsymbol{x}^{\max}$. При этом, согласно свойствам интервального анализа, q – это расстояние на интервальном множестве $\mathbb{IR}$ как целочисленных, так и вещественных интервалов (поскольку, если $\mathbb{R} \supset \mathbb{Z}$, то и $\mathbb{IR} \supset \mathbb{IZ}$). Отображение q на множестве $\mathbb{IR}$ задает метрику, которая является хаусдорфовой [132]. Хаусдорфова метрика обобщает понятие расстояния между

двумя точками в замкнутом метрическом пространстве (в данном контексте таким пространством является множество $\mathbb{R}$) на случай всех компактных непустых подмножеств данного пространства. При введении на множестве $\mathbb{IR}$ метрики оно становится топологическим пространством с сохранением понятий сходимости и непрерывности, как и в случае метрического пространства.

Точки $x_1^{\min}$ и $x_2^{\max}$ сформированы с позиции учета погрешности, вносимой в значения границ полем A тетракода. При этом максимальное значение q достигается для границ следующим образом:

- для левой границы: при минимальном значении поля M (все тетриты M сводятся к двоичному 0) значение поля A максимально (все тетриты A принимают значения двоичной 1);
- для правой границы: при максимальном значении поля M (все тетриты M сводятся к двоичной 1) значение поля A минимально (все тетриты A принимают значения двоичного 0).

Такая концепция предполагает определение параметров $\operatorname{wid} \boldsymbol{x}^{\max}$ и q, при которых представленные тетракодом границы интервала будут находиться в выделенных на рисунке 3.8, *б* участках числовой оси.

Поскольку на расстояние q оказывает влияние только значение, возвращаемое полем A тетракода, то его также можно выразить через позицию старшего тетрита поля A:

$$q = 2^{l+1} - 1, \tag{3.12}$$

где l – позиция старшего тетрита равного A в n-разрядном тетракоде: $k - 1 \le l \le 0$ (k – позиция старшего тетрита равного M).

При этом значение $\operatorname{wid} \boldsymbol{x}^{\min}$ можно определить, объединив соотношения (3.10) и (3.12):

$$\operatorname{wid} \boldsymbol{x}^{\min} = 2^{k+1} - 1 - 2 \cdot (2^{l+1} - 1) = 2^{k+1} - 2^{l+2} + 1. \tag{3.13}$$

Возможный диапазон измерения границ интервала, полученного при декодировании тетракода, обусловлен заполнением поля A случайными двоичными значениями и определяется следующим образом:

$$[\,x_i^{\min}\,,\,x_i^{\max}\,] = [\,x_i^{\max} - q\,,\,x_i^{\max}\,] = [\,x_i^{\min}\,,\,x_i^{\min} + q\,] \qquad (3.14)$$

при $i = 1$ – для левой границы; $i = 2$ – для правой границы.

В формулах (3.8), (3.12) и (3.13) используются переменные k и l, в которые подставляются номера позиций старших разрядов тетракода, равных M и A соответственно. Но иногда проще оперировать не позициями разрядов, а количеством тетритов в полях M и A.

Пусть u – количество тетритов M, а v — количество тетритов A в нормированном тетракоде. Тогда значения $\operatorname{wid}\boldsymbol{x}$, q, $\operatorname{wid}\boldsymbol{x}^{\max}$ и $\operatorname{wid}\boldsymbol{x}^{\min}$ определяются по формулам

$$\operatorname{wid}\boldsymbol{x} = 2^u - 1; \qquad (3.15)$$

$$q = 2^v - 1; \qquad (3.16)$$

$$\operatorname{wid}\boldsymbol{x}^{\max} = 2^{u+v} - 1; \qquad (3.17)$$

$$\operatorname{wid}\boldsymbol{x}^{\min} = 2^{u+v} - 2^{v+1} + 1. \qquad (3.18)$$

В приложении Г приведены расчеты всех метрик при декодировании тетракода 101MMAAA, а также показано получение предельных значений для границ результирующего интервала, т. е. границ интервалов $\boldsymbol{x}^{\min}$ и $\boldsymbol{x}^{\max}$ [80, с. 137]. Таким образом, при любой выборке (наступления события s_i) с учетом приведения поля A к случайному набору двоичных чисел, ширина интервала будет не больше $w_1 = 31$ и не меньше $w_2 = 17$. При этом диапазон возможного изменения границ интервала, декодированного из 101MMAAAt следующий: 160..167 для левой границы; 184..191 для правой границы.

Согласно таблице 3.1, тетракод 101MMAAA в памяти ЭВМ будет выглядеть следующим образом: 1001 1011 1100 0000b = 9BC0h.

При сведении n-разрядного нормированного тетракода, условно разделенного на поля знака, порядка и мантиссы, к вещественному интервалу, все оценки возможного размаха интервальных границ определяются аналогично описанным выше целочисленным представлениям. Однако сведение такого тетракода к двоичным значениям определяет формат последних, как формат с плавающей запятой. С учетом специфичности форматов чисел с плавающей

запятой, подобный тетракод может декодироваться в интервал, границы которого принадлежат области нормализованных (имеющих ненулевое значение в поле порядка) и денормализованных (имеющих нулевое значение в поле порядка) чисел с плавающей запятой [68, 118].

При этом для расчета ширины предполагаемых диапазонов, а также расстояния между ними, следует учитывать ряд дополнительных параметров, таких как десятичное значение поля порядка и его смещение, а также число разрядов мантиссы. Так, согласно [68, с. 165], для максимально «узких» и «широких» интервалов $\boldsymbol{x}'^{\min}$ и $\boldsymbol{x}'^{\max}$ и единственного интервала $\boldsymbol{x}'$ ширина wid определяется следующими соотношениями:

$$\operatorname{wid} \boldsymbol{x}' = \xi_{subnorm,\, norm} \cdot \operatorname{wid} \boldsymbol{x}; \tag{3.19}$$

$$\operatorname{wid} \boldsymbol{x}'^{\min} = \xi_{subnorm,\, norm} \cdot \operatorname{wid} \boldsymbol{x}^{\min}; \tag{3.20}$$

$$\operatorname{wid} \boldsymbol{x}'^{\max} = \xi_{subnorm,\, norm} \cdot \operatorname{wid} \boldsymbol{x}^{\max}, \tag{3.21}$$

где $\xi_{subnorm,\, norm}$ — коэффициент, учитывающий особенности формата представления чисел с плавающей запятой стандарта IEEE 754-2008 [54]: $\xi_{subnorm}$ — для денормализованных («субнормальных», *subnormal*) чисел; ξ_{norm} — для нормализованных (нормальных, *normal*) чисел.

Таким образом,

$$\xi_{subnorm} = 2^{1-offset-m}; \tag{3.22}$$

$$\xi_{norm} = 2^{E_{10}-offset-m}, \tag{3.23}$$

где E_{10} — десятичное значение поля порядка числа в формате с плавающей запятой, m — количество двоичных разрядов поля мантиссы формата; $offset = 2^{n-1} - 1$ — определенное стандартом IEEE 754 смещение значения порядка, занимающего поле из n двоичных разрядов.

И, наконец, расстояние q' между интервалами $\boldsymbol{x}'^{\min}$ и $\boldsymbol{x}'^{\max}$ можно получить, выразив через (3.11) значения $\operatorname{wid} \boldsymbol{x}^{\min}$ и $\operatorname{wid} \boldsymbol{x}^{\max}$ по (3.20) и (3.21):

$$q' = \frac{1}{2} \cdot (\operatorname{wid} \boldsymbol{x}'^{\max} - \operatorname{wid} \boldsymbol{x}'^{\min}) = \xi_{subnorm,\, norm} \cdot q. \tag{3.24}$$

Следует отметить, что зависимость (3.19) применима для нормированного тетракода, содержащего только поле M, а зависимости (3.20) и (3.21) — для нормированного тетракода с полями M и A.

Все правила нормированности для тетракода, содержащего в себе вещественный интервал, относятся только к полям мантиссы и не предполагают появления неоднозначно представимых значений тетритов в поле порядка. В противном случае тетракод может декодироваться к настолько «широкому» интервалу, который уже не представляет никакой информационной значимости.

В приложении Г приведены все расчеты при декодировании 32-разрядного тетракода к интервалу, границы которого заданы в формате с плавающей запятой одинарной точности [80, с. 141]. Таким образом, при максимально возможном разбросе границ интервала q', приближенно равным $3{,}8\cdot10^{-6}$, для любой выборки с учетом возвращения полем A случайного набора двоичных чисел, ширина интервала не превысит значение $6{,}102\cdot10^{-5}$ и не станет меньше, чем $5{,}342\cdot10^{-5}$. При этом диапазон возможного изменения границ интервала, декодированного из данного тетракода следующий: 0,21441650..0,21442030 для левой границы; 0,21447372..0,21447752 для правой границы.

В памяти бинарного компьютера приведенный в приложении Г тетракод, согласно таблице 3.1, будет представлен следующей последовательностью бит:

0101101010101001011001101001101010010110111111110000000000000000,

или 5AA9 669A 96FF 0000h – в сокращенной hex-записи.

При декодировании тетракода к границам интервала, лежащего в отрицательной области вещественной оси необходимо учитывать особенность отрицательных чисел: числа с большим модулем находятся на отрицательной части числовой оси левее, чем числа с меньшим модулем.

Методика оценивания границ вещественного интервала при декодировании тетракода не зависит от знакового разряда чисел, если в тетракоде он определен однозначно. В этом случае у результирующего интервала границы одного знака, т. е интервал занимает позицию либо на положительной, либо на отрицательной

полуосях вещественных чисел. В работе [68, с. 168–171] рассмотрен случай декодирования тетракода, содержащего в знаковом разряде значение М. Такой тетракод отвечает обозначенным выше требованиям к нормированности (относящимся, прежде всего, к полю мантиссы). При формировании значения левой границы тетрит М в поле знака приводится к двоичной 1 (формируется отрицательное значение), а при формировании правой границы – к двоичному 0 (формируется положительное значение). Полученный при этом результирующий интервал гарантировано содержит нулевое значение. Более того, он фактически является интервальным окружением нуля. На рисунке 3.9 показана «разметка» интервальных границ вещественной оси для интервалов, полученных из одного тетракода, но с разными значениями тетрита в знаковом разряде: 0 ($\boldsymbol{x}' \geq 0$), 1 ($\boldsymbol{x}' < 0$) и М ($\boldsymbol{x}'' \supset 0$).

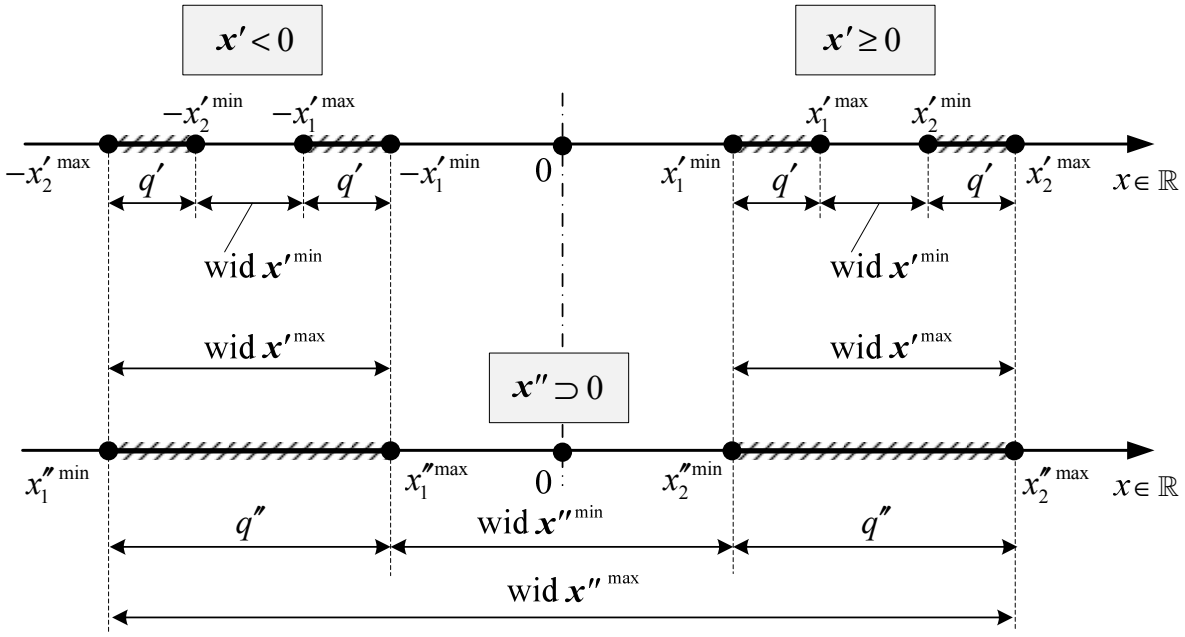


Рисунок 3.9 – Формирование вещественных интервальных границ для тетракода при значениями знакового разряда, равного 0 ($\boldsymbol{x}' \geq 0$), 1 ($\boldsymbol{x}' < 0$) и М ($\boldsymbol{x}'' \supset 0$).

В приложении Д показаны варианты получения значений границ результирующих максимального (т. е. имеющего максимально возможное значение ширины) интервала (скриншот *а*), минимального (т. е. имеющего

минимально возможное значение ширины) интервала (скриншот *б*), а также и одного из множества возможных (поле A заполнено случайными двоичными значениями) интервалов (скриншот *в*) при декодировании «вещественного» тетракода. При этом очевидно, что в любой из возможных результирующих интервалов будут включены все значения минимального интервала. В последующем, при выполнении арифметических операций над тетракодами анализу будет подвергаться обязательное сохранение числовых значений, принадлежащих минимальному интервалу.

При программно-аппаратной реализации декодировании тетракода, т. е. при поразрядной замене тетритов t_i (i – позиция тетрита в n-разрядном тетракоде, $n-1 \le i \le 0$) эквивалентными двоичными значениями, можно прибегнуть к предложенным в п. 2.2 (Таблица 2.1) унарным функциям тетралогики: MIN_A (MAX_A) для минимизации (максимизации) неопределенности, т. е. значения A в тетракоде; MIN_M (MAX_M) – минимизации (максимизации) множественности, т. е. значение M в тетракоде. Используя сочетания этих функций, можно все значения тетритов A и M привести к значениям 0 или 1. Тогда тетракод будет состоять только из тетритов 0 и 1, т. е. фактически представлять двоичное значение одной из интервальных границ.

При сведении n-разрядного тетракода к значениям интервальных границ используются следующие сочетания:

$$\mathrm{MIN\_M}(\mathrm{MIN\_A}(t_i)) = \begin{cases} 0, & if\ t_i = 0 \vee t_i = \mathrm{A} \vee t_i = \mathrm{M}, \\ 1, & if\ t_i = 1; \end{cases} \tag{3.25}$$

$$\mathrm{MIN\_M}(\mathrm{MAX\_A}(t_i)) = \begin{cases} 0, & if\ t_i = 0 \vee t_i = \mathrm{M}, \\ 1, & if\ t_i = 1 \vee t_i = \mathrm{A}; \end{cases} \tag{3.26}$$

$$\mathrm{MAX\_M}(\mathrm{MIN\_A}(t_i)) = \begin{cases} 0, & if\ t_i = 0 \vee t_i = \mathrm{A}, \\ 1, & if\ t_i = 1 \vee t_i = \mathrm{M}; \end{cases} \tag{3.27}$$

$$\mathrm{MAX\_M}(\mathrm{MAX\_A}(t_i)) = \begin{cases} 0, & if\ t_i = 0, \\ 1, & if\ t_i = 1 \vee t_i = \mathrm{A} \vee t_i = \mathrm{M}. \end{cases} \tag{3.28}$$

Таким образом, при определении численных значений интервальных границ можно проследить тождественность выражений (3.25)–(3.28) и записей преобразований, приведенных в приложении Г:

$$\mathrm{M} \to 0 \wedge \mathrm{A} \to 0 \equiv \mathrm{MIN\_M}(\mathrm{MIN\_A}(t_i));$$

$$\mathrm{M} \to 0 \wedge \mathrm{A} \to 1 \equiv \mathrm{MIN\_M}(\mathrm{MAX\_A}(t_i));$$

$$\mathrm{M} \to 1 \wedge \mathrm{A} \to 0 \equiv \mathrm{MAX\_M}(\mathrm{MIN\_A}(t_i));$$

$$\mathrm{M} \to 1 \wedge \mathrm{A} \to 1 \equiv \mathrm{MAX\_M}(\mathrm{MAX\_A}(t_i)).$$

Используя соотношения (3.12)–(3.24), можно провести интервальное оценивание тетракода и получить необходимые метрики результирующего интервального числа, не прибегая к декодированию двоичных или десятичных значений интервальных границ. При этом эффективное хранение интервального числа в тетракоде позволяет оперировать интервальными объектами в постбинарных компьютерных системах.

3.5 Основные свойства арифметических операций над тетракодами

Поскольку тетракод $\mathrm{C}_4 = \{0, \mathrm{A}, \mathrm{M}, 1\}$ используется для хранения числовой информации, над ним могут быть определены арифметические операции.

Пусть множество $\mathrm{o}\rho = \{+, -, \times, \div\}$ является множеством арифметических операций над тетракодами. Тогда для тетракодов T_1 и T_2 определена:

- операция сложения, имеющая запись $T_1 + T_2$;
- операция вычитания, записывающаяся как $T_1 - T_2$;
- операция умножения, имеющая запись $T_1 \times T_2$;
- операция деления – $T_1 \div T_2$, при соблюдении условия $0 \not\subset T_2$ (совокупность значений, декодированных из T_2, не содержит ноль).

Для операций $\mathrm{o}\rho$ справедливо большинство свойств аналогичных арифметических операций, определенных над действительными числами. Однако, ввиду «интервальной природы» тетракода (результатом декодирования тетракода в большинстве случаев является интервал), операции $\mathrm{o}\rho$ также должны

соответствовать свойствам определенных над интервалами арифметических операций: (выражения (1.10)–(1.13) и рисунок 1.3). Порядок выполнения операций ρ аналогичен порядку выполнения одноименных арифметических операций, принятому в математике.

В частности, справедливость каждой из арифметических операций над тетракодами основана на следующих соответствиях:

1) Соответствие сложения и вычитания:

- $T_1 + T_2$ равно T_3 тогда и только тогда, когда $T_3 - T_2 = T_1$ и $T_3 - T_1 = T_2$;
- $T_1 - T_2 = T_3$ тогда и только тогда, когда $T_3 + T_2 = T_1$ и $T_1 - T_3 = T_2$;

2) Соответствие умножения и деления:

- $T_1 \times T_2 = T_3$ тогда и только тогда, когда $T_3 \div T_2 = T_1$ и $T_3 \div T_1 = T_2$;
- $T_1 \div T_2 = T_3$ тогда и только тогда, когда $T_3 \times T_2 = T_1$ и $T_1 \div T_3 = T_2$;

Также для операций сложения и умножения из ρ обязательно выполнение свойств коммутативности

$$T_1 + T_2 = T_2 + T_1; \tag{3.29}$$

$$T_1 \times T_2 = T_2 \times T_1, \tag{3.30}$$

и ассоциативности

$$(T_1 + T_2) + T_3 = T_1 + (T_2 + T_3) = T_1 + T_2 + T_3; \tag{3.31}$$

$$(T_1 \times T_2) \times T_3 = T_1 \times (T_2 \times T_3) = T_1 \times T_2 \times T_3. \tag{3.32}$$

Аналогично интервальной арифметике, для операций ρ свойство дистрибутивности может иметь место в ослабленном виде [133]:

$$T_1 \times (T_2 + T_3) \subset T_1 \times T_2 + T_1 \times T_3. \tag{3.33}$$

Выражение (3.33) указывает о субдистрибутивности умножения относительно сложения. Однако при дальнейшей разработке арифметических операций над тетракодами подтвердился факт наличия строгой дистрибутивности умножения относительно сложения.

3.5.1 Сложение тетракодов

Сложение двух тетракодов аналогично сложению результирующих интервалов, декодированных из данных тетракодов-операндов.

На рисунке 3.10 представлены два интервала, полученные при приведении тетракода T к двоичному коду. При этом значения границ каждого из интервалов получены приведением тетритов тетракода к битам двоичного кода (обозначено знаком «$\rightarrow$»). Это позволяет заменить операцию сложения тетритов двоичным сложением: $0+0=0$, $0+1=1$ и $1+1=10$.

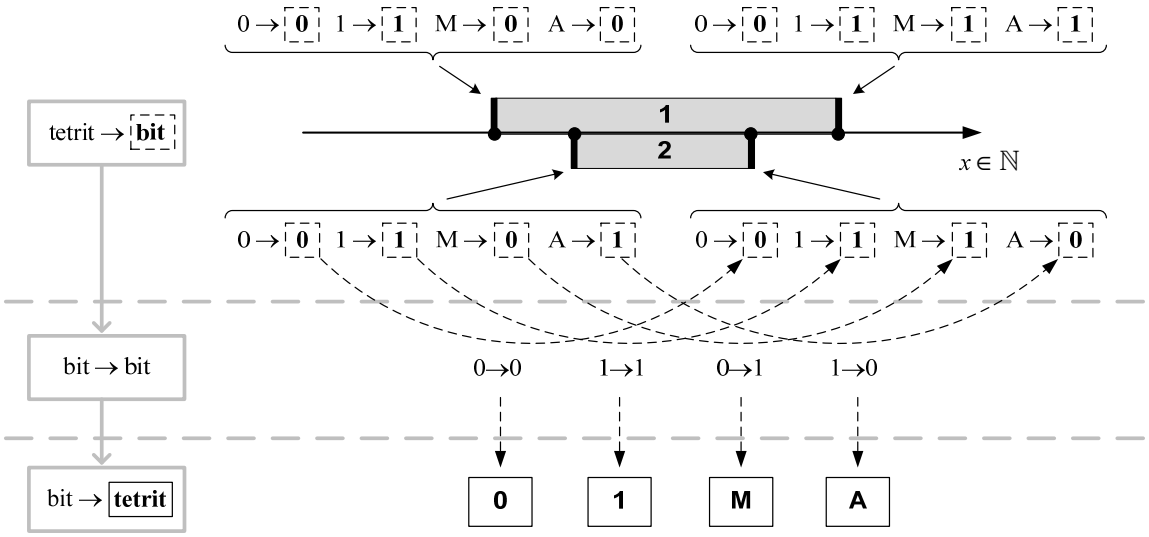


Рисунок 3.10 – Битовое (bit) представление тетритов (tetrit) в тетракоде по отношению к границам интервалов с максимальной (интервал № 1) и минимальной (интервал № 2) шириной

Интервал 2 является минимальным интервалом, т. е. интервалом с минимально возможной шириной (Рисунок 3.8), поэтому все его значения будут принадлежать всем возможным интервалам, полученным при декодировании тетракода T. Если интервал $\boldsymbol{t}_1^{\min}$ является минимальным интервалом, декодированным из тетракода T_1, а $\boldsymbol{t}_2^{\min}$ – минимальный интервал из T_2, то результатом сложения $T_1 + T_2$ является такой тетракод T_3, при декодировании которого полученный минимальный интервал $\boldsymbol{t}_3$ представляет собой сумму

интервалов $\boldsymbol{t}_1^{\min}$ и $\boldsymbol{t}_2^{\min}$ [134]. Следует отметить, что при любых значениях двух тетракодов всегда найдется третий тетракод, являющийся суммой двух предыдущих:

$$\forall\ T_1 \to \boldsymbol{t}_1^{\min},\ T_2 \to \boldsymbol{t}_2^{\min}\ \ \exists\ T_3 \to \boldsymbol{t}_3^{\min}:\ T_1 + T_2 = T_3 \wedge \boldsymbol{t}_1^{\min} + \boldsymbol{t}_2^{\min} = \boldsymbol{t}_3^{\min}. \qquad (3.34)$$

Запись (3.34) также указывает на то, что сложение тетракодов аналогично сложению интервалов.

При переходе от меньшей границы этого интервала к большей, наблюдаются переходы двоичных значений от 0 к 1 на месте тетритов М и от 1 к 0 на месте тетритов А (обозначены на рисунке 3.10 изогнутыми пунктирными линиями). Данный аспект позволяет произвести обратный переход от двоичного сложения к сложению тетракодов и сформулировать алгоритм выполнения операции сложения двух тетритов:

Шаг 1. Замена тетритов битами в соответствии с рисунком 3.10 для интервала № 2 и формирование интервалов, имеющих двоичные значения границ.

Шаг 2. Выполнение операции сложения над сформированными интервалами по формуле (1.10), и получение двоичных значений границ результирующего интервала.

Шаг 3. Путем побитового сопоставления левой и правой границ результирующего интервала формируется один или пара тетритов (сумма и перенос в старший разряд) по следующим критериям (следуют из рисунка 3.10):

- если значение сопоставленных битов не изменилось, то результатом является эквивалентный биту тетрит: $(\textit{бит}\,0 \to \textit{бит}\,0) \to \textit{тетрит}\,0$, $(\textit{бит}\,1 \to \textit{бит}\,1) \to \textit{тетрит}\,1$;
- если значение сопоставленных битов изменилось, то результатом является тетрит М или А: $(\textit{бит}\,0 \to \textit{бит}\,1) \to \textit{тетрит}\,\text{М}$, $(\textit{бит}\,1 \to \textit{бит}\,0) \to \textit{тетрит}\,\text{А}$.

В левой части рисунка 3.10 фактически показано поэтапное выполнение приведенного выше алгоритма сложения.

В приложении Е показано получение суммы для некоторых пар тетритов [66, с. 97–104], а в таблице 3.2 сведены результаты сложений тетритов во всех возможных сочетаниях. Из схематических изображений в приложении Е видно, что сложение для тетритов коммутативно, поскольку коммутативно двоичное сложение.

Сложение тетракодов выполняется поразрядно, начиная с младшего разряда и используя значения таблице 3.2. В приложении Ж (Рисунок Ж.1) приведены два примера сложения тетракодов с демонстрацией проверки правильности полученной суммы путем сложения минимальных интервалов и сопоставления полученных результатов.

Таблица 3.2 – Таблица сложения для тетракодов

+	0	A	M	1
0	0	A	M	1
A	A	A0	1	AM
M	M	1	M0	MA
1	1	AM	MA	10

Поскольку сложение интервальных чисел обладает свойствами ассоциативности, сложение тетракодов также не противоречит свойству (3.31). Это также подтверждается примерами, показывающими справедливость ассоциативности сложения тетритов:

$$1+1+\mathrm{A}=\begin{vmatrix}(1+1)+\mathrm{A}=10+\mathrm{A}=1\mathrm{A},\\ 1+(1+\mathrm{A})=1+\mathrm{AM}=1\mathrm{A};\end{vmatrix}$$

$$1+1+\mathrm{M}=\begin{vmatrix}(1+1)+\mathrm{M}=10+\mathrm{M}=1\mathrm{M},\\ 1+(1+\mathrm{M})=1+\mathrm{MA}=1\mathrm{M};\end{vmatrix}$$

$$1+\mathrm{A}+\mathrm{M}=\begin{vmatrix}(1+\mathrm{A})+\mathrm{M}=\mathrm{AM}+\mathrm{M}=10,\\ 1+(\mathrm{A}+\mathrm{M})=1+1=10;\end{vmatrix}$$

$$A + A + M = \begin{vmatrix} (A + A) + M = A0 + M = AM, \\ A + (A + M) = A + 1 = AM; \end{vmatrix}$$

$$M + M + A = \begin{vmatrix} (M + M) + A = M0 + A = MA, \\ M + (M + A) = M + 1 = MA. \end{vmatrix}$$

В таблице 3.3 показана связность полученных результатов сложения тетритов из таблицы 3.2 с результатами операции суммы по модулю 2 тетралогики из таблицы 2.7. Действительно, младшие разряды результатов сложения тетритов-слагаемых совпадают с результатами суммы по модулю 2 этих же пар тетритов.

Таблица. 3.3 – Сопоставление результатов при арифметической операции сложения и логической операции суммы по модулю 2

Арифметика

+	0	A	M	1
0	0	A	M	1
A	A	A0	1	AM
M	M	1	M0	MA
1	1	AM	MA	10

Логика

⊕	0	A	M	1
0	0	A	M	1
A	A	0	1	M
M	M	1	0	A
1	1	M	A	0

3.5.2 Вычитание тетракодов

Согласно соответствию сложения и вычитания для тетракодов ($T_1 - T_2 = T_3$ тогда и только тогда, когда $T_3 + T_2 = T_1$ и $T_1 - T_3 = T_2$), операцию вычитания, а именно результат вычитания тетритов 0, A, M и 1 во всех сочетаниях, можно получить из таблицы сложения для тетракодов (Таблица 3.2). Результат приведен в таблице 3.4 (в скобках указан заем из старшего разряда).

Поскольку операция вычитания фактически получена из результатов операции сложения, то и для вычитания справедливы следующие утверждения:

1) вычитание тетритов основывается на вычитании двоичных разрядов;
2) вычитание двух тетракодов аналогично вычитанию результирующих интервалов, декодированных из данных тетракодов-операндов.

Таблица 3.4 – Таблица вычитания для тетракодов

t_1	t_2	$t_1 - t_2$	t_1	t_2	$t_1 - t_2$	t_1	t_2	$t_1 - t_2$	t_1	t_2	$t_1 - t_2$
0	0	(0)0	A	0	(0)A	M	0	(0)M	1	0	(0)1
	A	(A)A		A	(0)0		A	(A)1		A	(0)M
	M	(M)M		M	(M)1		M	(0)0		M	(0)A
	1	(1)1		1	(M)M		1	(A)A		1	(0)0

Эти утверждения делают возможной проверку вычитания тетракодов путем вычитания по формуле (1.11) минимальных (имеющих минимально возможное значение ширины) интервалов, декодированных из тетракодов-операндов. Аналогично записи (3.34) для сложения, при вычитании тетракодов справедлива следующая запись:

$$\forall\ T_1 \to \boldsymbol{t}_1^{\min},\ T_2 \to \boldsymbol{t}_2^{\min}: \boldsymbol{t}_1^{\min} \geq \boldsymbol{t}_2^{\min}\ \exists\ T_3 \to \boldsymbol{t}_3:$$
$$T_1 - T_2 = T_3 \wedge \boldsymbol{t}_1^{\min} - \boldsymbol{t}_2^{\min} \supseteq \boldsymbol{t}_3. \qquad (3.35)$$

Выражение (3.35) показывает, что если интервал $\boldsymbol{t}_1^{\min}$ является минимальным интервалом, декодированным из тетракода T_1, а $\boldsymbol{t}_2^{\min}$ – минимальным интервалом из T_2, причем $\boldsymbol{t}_1^{\min} \geq \boldsymbol{t}_2^{\min}$, то разностью $T_1 - T_2$ является такой тетракод T_3, при декодировании которого полученный минимальный интервал $\boldsymbol{t}_3$ представляет собой подинтервал интервала, полученного при $\boldsymbol{t}_1^{\min} - \boldsymbol{t}_2^{\min}$. Запись $\boldsymbol{t}_1^{\min} - \boldsymbol{t}_2^{\min} \supseteq \boldsymbol{t}_3$ показывает, что в результате вычитания тетракодов сохраняется достоверность интервального результата [134].

Вычитание тетракодов выполняется поразрядно, начиная с младшего разряда, используя при этом значения таблицы 3.4. В приложении Ж

(Рисунок Ж.2) приведены два примера вычитания тетракодов с проверкой правильности полученных результатов.

В архитектуре постбинарной ЭВМ, также как и в архитектуре современной (бинарной) ЭВМ реализация вычитающих схем является избыточной. Поэтому для упрощения архитектуры ЭВМ можно заменить операцию вычитания тетракодов на операцию сложения. При этом наиболее распространенным способом представления отрицательных чисел в двоичном коде является использование дополнительного кода – величины, полученной вычитанием числа из наибольшей степени двух (из 2^n для n-битного дополнения до 2).

Так операцию вычитания двух двоичных кодов B_1 и B_2 можно заменить операцией сложения следующим образом:

$$B_1 - B_2 = B_1 + (-B_2) = B_1 + (B_2)_{\text{ДК}}, \tag{3.36}$$

где запись $(B_2)_{\text{ДК}}$ подразумевает значение операнда B_2 в дополнительном коде.

Поскольку сложение и вычитание тетритов производится аналогично сложению и вычитанию битов, то для тетракодов T_1 и T_2 равенство (3.36) можно записать следующим образом:

$$T_1 - T_2 = T_1 + (-T_2) = T_1 + (T_2)_{\text{ДК}}, \tag{3.37}$$

где запись $(B_2)_{\text{ДК}}$ подразумевает значение операнда B_2 в дополнительном коде.

Дополнительный код тетракода имеет тот же смысл, что и в двоичном коде (т. е. дополнение до 2, англ. *two's complement*), и его можно получить поразрядным инвертированием тетракода (первое дополнение) и прибавлением к инверсии тетрита со значением 1 (второе дополнение), либо вычитанием тетракода из нуля. Первое дополнение тетракода (дополнение до 1, англ. *ones' complement* [135]) называется обратным кодом тетракода.

Заметим, что при получении дополнительного кода тетракода T, в качестве логической инверсии и арифметического сложения применяются те операции, которые определены для тетракодов:

$$T_{\text{ДК}} = \overline{T} + 1\text{t}.$$

Так, для примера 1 рисунок Ж.2, вычитание

10010111t – 10MMAAt = 0.10010111t – 0.0010MMAAt = 0.011AMAMMt

можно заменить операцией сложения в дополнительном коде

$$0.10010111\mathrm{t} + (1.1101\mathrm{AAMMt} + 1\mathrm{t}) = 0.10010111\mathrm{t} + 1.1101\mathrm{A10At} = 0.011\mathrm{AMAMMt}.$$

3.5.3 Умножение и деление тетракодов

Для умножения тетракодов действуют правила, которые полностью совпадают с аналогичными, применяемыми к двоичным числам. Так для тетракода T действуют правила:

- умножения на нуль: $T \times 0\mathrm{t} = 0\mathrm{t}$;
- умножения на единицу: $T \times 1\mathrm{t} = T$;
- умножения при равных множителях: $T \times T = T$.

В таблице 3.5 приведены результаты умножения двух тетритов, а в таблице 3.6 – идентичность полученных результатов с результатами операции умножения тетралогики из таблицы 2.2.

Умножение тетракодов, также как и сложение, эквивалентно интервальному умножению, о чем свидетельствует запись (3.37). Она показывает, что если интервал $\boldsymbol{t}_1$ является минимальным интервалом, декодированным из тетракода T_1, а $\boldsymbol{t}_2$ – минимальным интервалом из T_2, то результатом умножения $T_1 \times T_2$ является такой тетракод T_3, при декодировании которого полученный минимальный интервал $\boldsymbol{t}_3$ представляет собой произведению интервалов $\boldsymbol{t}_1^{\min} \cdot \boldsymbol{t}_2^{\min}$. Произведение интервалов выполняется по формуле (1.12). Следует отметить, что при любых значениях двух тетракодов всегда найдется третий тетракод, являющийся произведением двух предыдущих:

$$\forall\ T_1 \to \boldsymbol{t}_1^{\min},\ T_2 \to \boldsymbol{t}_2^{\min}\ \ \exists\ T_3 \to \boldsymbol{t}_3^{\min}:\ T_1 \times T_2 = T_3 \wedge\ \boldsymbol{t}_1^{\min} \cdot \boldsymbol{t}_2^{\min} = \boldsymbol{t}_3^{\min}. \qquad (3.38)$$

Таблица 3.5 – Таблица умножения для тетракодов

×	0	A	M	1
0	0	0	0	0
A	0	A	0	A
M	0	0	M	M
1	0	A	M	1

Таблица 3.6 – Идентичность результатов при арифметическом и логическом умножении

Арифметика

×	0	A	M	1
0	0	0	0	0
A	0	A	0	A
M	0	0	M	M
1	0	A	M	1

Логика

∧	0	A	M	1
0	0	0	0	0
A	0	A	0	A
M	0	0	M	M
1	0	A	M	1

В приложении Ж (Рисунок Ж.3) показаны два примера умножения тетракодов с демонстрацией проверки правильности полученного произведения через умножение интервальных значений с последующим постбинарным кодирования интервала-результата.

Поскольку умножение интервальных чисел обладает свойством ассоциативности, умножение тетракодов также не противоречит свойству (3.32), что подтверждается следующими примерами:

$$1\times 1\times \mathrm{A} = \left| \begin{array}{l} (1\times 1)\times \mathrm{A} = 1\times \mathrm{A} = \mathrm{A}, \\ 1\times (1\times \mathrm{A}) = 1\times \mathrm{A} = \mathrm{A}; \end{array} \right.$$

$$1\times 1\times \mathrm{M} = \left| \begin{array}{l} (1\times 1)\times \mathrm{M} = 1\times \mathrm{M} = \mathrm{M}, \\ 1\times (1\times \mathrm{M}) = 1\times \mathrm{M} = \mathrm{M}; \end{array} \right.$$

$$1\times \mathrm{A}\times \mathrm{M} = \left| \begin{array}{l} (1\times \mathrm{A})\times \mathrm{M} = \mathrm{A}\times \mathrm{M} = 0, \\ 1\times (\mathrm{A}\times \mathrm{M}) = 1\times 0 = 0; \end{array} \right.$$

$$\mathrm{A}\times \mathrm{A}\times \mathrm{M} = \left| \begin{array}{l} (\mathrm{A}\times \mathrm{A})\times \mathrm{M} = \mathrm{A}\times \mathrm{M} = 0, \\ \mathrm{A}\times (\mathrm{A}\times \mathrm{M}) = \mathrm{A}\times 0 = 0; \end{array} \right.$$

$$\mathrm{M}\times \mathrm{M}\times \mathrm{A} = \left| \begin{array}{l} (\mathrm{M}\times \mathrm{M})\times \mathrm{A} = \mathrm{M}\times \mathrm{A} = 0, \\ \mathrm{M}\times (\mathrm{M}\times \mathrm{A}) = \mathrm{M}\times 0 = 0. \end{array} \right.$$

Полученные сложение и умножение тетракодов обладает важнейшим свойством дистрибутивности (в отличие от предполагаемой в (3.33) субдистрибутивности) умножения относительно сложения. Например,

$$\mathrm{M}\times (1+\mathrm{A}) = \left| \begin{array}{l} \mathrm{M}\times \mathrm{AM} = \mathrm{M}, \\ \mathrm{M}\times 1 + \mathrm{M}\times \mathrm{A} = \mathrm{M} + 0 = \mathrm{M}; \end{array} \right.$$

$$\mathrm{A}\times (\mathrm{M}+1) = \left| \begin{array}{l} \mathrm{A}\times \mathrm{MA} = \mathrm{A}, \\ \mathrm{A}\times \mathrm{M} + \mathrm{A}\times 1 = 0 + \mathrm{A} = \mathrm{A}. \end{array} \right.$$

Аналогично двоичному умножению, при умножении произвольного тетракода T на тетракод, содержащий число вида 2^k, $k \in \mathbb{N}$, произведение можно получить путем сдвига разрядов тетракода T влево на k разрядов с заполнением освободившихся разрядов значением 0.

Деление тетракодов также аналогично делению двоичных чисел. При организации деления «в столбик» используются операции умножения и вычитания тетракодов. Учитывая рассмотренную в п. 3.5 связь между умножением и делением, тетракоды из примеров (Рисунок Ж.3) можно связать операцией деления (Рисунок Ж.4).

При делении тетракодов справедлива следующая запись

$$\forall\ T_1 \to \boldsymbol{t}_1^{\min},\ T_2 \to \boldsymbol{t}_2^{\min}:\ \exists\ T_3 \to \boldsymbol{t}_3:\ T_1 \div T_2 = T_3 \wedge \boldsymbol{t}_1^{\min} / \boldsymbol{t}_2^{\min} \supseteq \boldsymbol{t}_3, \qquad (3.39)$$

которая показывает, что если интервал $\boldsymbol{t}_1^{\min}$ является минимальным интервалом, декодированным из тетракода T_1, а $\boldsymbol{t}_2^{\min}$ – минимальным интервалом из T_2, причем $\boldsymbol{t}_1^{\min} \geq \boldsymbol{t}_2^{\min}$, то частным $T_1 \div T_2$ является такой тетракод T_3, при декодировании которого полученный минимальным интервал $\boldsymbol{t}_3$ представляет собой подинтервал интервала, полученного при делении $\boldsymbol{t}_1^{\min}$ на $\boldsymbol{t}_2^{\min}$. Запись $\boldsymbol{t}_1^{\min} / \boldsymbol{t}_2^{\min} \supseteq \boldsymbol{t}_3$

показывает, что в результате деления тетракодов сохраняется достоверность интервального результата.

Аналогично двоичному делению, при делении произвольного тетракода T на тетракод, содержащий число вида 2^k, $k \in \mathbb{N}$, частное можно получить путем сдвига разрядов тетракода T вправо на k разрядов с заполнением освободившихся разрядов значением 0.

3.6 Выводы по третьей главе

Рассмотренные принципы постбинарного кодирования и декодирования лежат в основе компьютерных вычислений, в которых используются функции тетралогики и принципы тетракодирования.

Полученные в пунктах 3.1–3.4 результаты кодирования и декодирования тетракодов, а также приведенные в п. 3.4 зависимости и соотношения позволяют сформировать следующие заключения:

1. Кодирование числовой информации с помощью четырехзначноей кодовой системы {0, A, M, 1} эффективно только при использовании основных принципов тетракдирования, сформулированных в п. 3.1.

2. Применительно к тетракодам справедливы следующие действия: кодирование, декодирование и представление. Определения каждого действия даны в п. 3.1, а сам порядок их проведения представлены в пп. 3.2–3.4.

3. Использование тетракодирования целесообразно при хранении с последующим применением в вычислениях одной или нескольких количественных характеристик, являющихся конечным подмножеством множества значений одного типа данных.

4. Декодирование «целочисленных» тетракодов приводит к получению одного или множества значений (в частности, границ интервала) целого типа, являющихся конечным подмножеством бесконечного множества целых чисел, ограниченного максимальным и минимальным значениями.

5. Декодирование «вещественных» тетракодов приводит к получению одного или нескольких значений (в частности, границ интервала) множества действительных чисел, представленных в форматах с плавающей запятой стандарта IEEE 754-2008.

6. Для точного получения границ результирующего интервала при декодировании тетракода необходимо соблюдение так называемой «нормированности» последнего, т. е. такое упорядочивание значений M и A, чтобы возможно было получение значений границ максимального и минимального интервалов.

7. Для проведения «интервального оценивания» тетракода отработан способ расчета диапазона изменения («плавания») границ без численного определения всех вероятных границ для результирующего интервала.

8. При сведении тетракода к значениям интервальных границ используются сочетания функций минимизации и максимизации тетралогики для значений M и A.

9. При «интервальном оценивании» тетракода все значения, лежащие внутри границ «крайне узкого» интервала также гарантированно принадлежат результирующему интервалу, полученному при декодировании тетракода.

На основании полученных принципов кодирования и декодирования тетракодов и приведения последних к интервальным значениям, разработана и обоснована тетраарифметика – множество взаимосвязанных операций сложения, вычитания, умножения и деления тетракодов, с соответствующим подтверждением выполнения основных свойств полученных арифметических операций.

РАЗДЕЛ 4
МОДИФИКАЦИЯ СТАНДАРТНЫХ ФОРМАТОВ ЧИСЕЛ С ПЛАВАЮЩЕЙ ЗАПЯТОЙ

4.1 Особенности модификации стандартных форматов чисел с плавающей запятой

Основные недостатки современного представления чисел с плавающей запятой заключаются в следующем:

– большинство чисел вещественной оси не могут быть представлены точно числами с плавающей запятой, имеющими конечную длину мантиссы, и, соответственно, свойства арифметических операций над числами с плавающей запятой отличаются (из-за неизбежных округлений) от свойств идеальных математических операций над вещественными числами;

– число в формате с плавающей запятой не несет никакой информации о точности той величины, которую оно представляет.

Получается, что существующая модель вычислений с плавающей запятой не предназначена ни для адекватного представления исходных значений, ни для отслеживания вычислительных ошибок. В связи с этим постепенно усиливается тенденция к переходу от точечных значений к интервальным.

При этом предлагается ряд способов для преодоления проблем, связанных с ограничением разрядности чисел, поскольку использование при вычислениях разрядностей, существенно превышающих стандартную, также является одним из способов получения правильных результатов. Все эти способы в совокупности позволяют в процессе вычислений выполнять следующие операции [57]:

– увеличение (или выравнивание) разрядности во избежание переполнения разрядов результата и выполнения корректных вычислений;

– выполнение отложенного деления, когда отдельно вычисляются числитель и знаменатель, а деление производится на последнем шаге вычисления;

– организацию машинной интервальной арифметики.

В рамках реализации вышеперечисленных операций возможно достижение достаточно надежного и эффективного контроля корректности вычислений. Сами же операции можно эффективно использовать при разработке и реализации постбинарных методов вычислений, основанных на постбинарном представлении количественных значений [136, 137]. Однако введение арифметических операций в постбинарный вычислительный процесс невозможно без незначительных модификаций самих форматов чисел с плавающей запятой.

Согласно стандарту IEEE 754-2008 [54], машинным числом с плавающей запятой называют число вида:

$$x_{m,k} = \pm 0, s_1 s_2 ... s_m \cdot b^e = \pm \left(\frac{s_1}{b^1} + \frac{s_2}{b^2} + ... \frac{s_m}{b^m} \right) \cdot b^e = \pm \mu \cdot b^e, \quad (4.1)$$

где $b \in \{0, 1, ..., b-1\}$ – основание системы счисления (для бинарного компьютигна $b = 2$); μ – мантисса числа; m – длина или размер мантиссы (количество знаков в мантиссе); $s_i \in \{0, 1, ..., b-1\}$, $i = \overline{1, m}$ – значащие цифры ($s_1 \neq 0$); e – порядок числа; k – размер прядка (количество знаков в представлении порядка).

Машинное число (4.1) можно представить в виде *денормализованного числа* (англ. *denormalized numbers*, *subnormal numbers*), которое в машинной арифметике используется наряду *с нормализованными числами* с плавающей запятой.

Деормализованным числом с плавающей запятой называют число вида:

$$x'_{m,k} = \pm 0, s_1 s_2 ... s_m \cdot b^e = \pm \left(\frac{s_1}{b^1} + \frac{s_2}{b^2} + ... \frac{s_m}{b^m} \right) \cdot b^e = \pm 0, \mu' \cdot b^e, \quad (4.2)$$

а нормализованным числом с плавающей запятой – число вида:

$$x_{m,k} = \pm 1, s_1 s_2 ... s_m \cdot b^e = \pm \left(1 + \frac{s_1}{b^1} + \frac{s_2}{b^2} + ... \frac{s_m}{b^m} \right) \cdot b^e = \pm 1, \mu' \cdot b^e, \quad (4.3)$$

где $b \in \{0, 1, ..., b-1\}$ – основание системы счисления; μ' – остаток от мантиссы числа (мантисса без целой части); m – длина или размер мантиссы μ'; $s_i \in \{0, 1, ..., b-1\}$, $i = \overline{1, m}$ – значащие цифры дробной части числа.

В стандарте IEEE 754-2008 форматы чисел с плавающей запятой обозначают binaryΩ (Рисунок 4.1), где Ω/32 – точность формата (например,

binary16 – формат с половинной точностью, binary32 – с одинарной точностью, binary128 – с четверной точностью). Фактически справедливо тождество $\Omega \equiv n$, где n – длина или размер (количество знаков) формата числа.

На основании форматов чисел binaryΩ стандарта IEEE754-2008 предлагаются *модифицированные форматы чисел* pbinaryΩ различной точности (Рисунок 4.1): от одинарной ($n = 32$) до восьмерной ($n = 256$) [80, с. 64], [138, 139, 140, 141, 142, 143, 144, 145, 146].

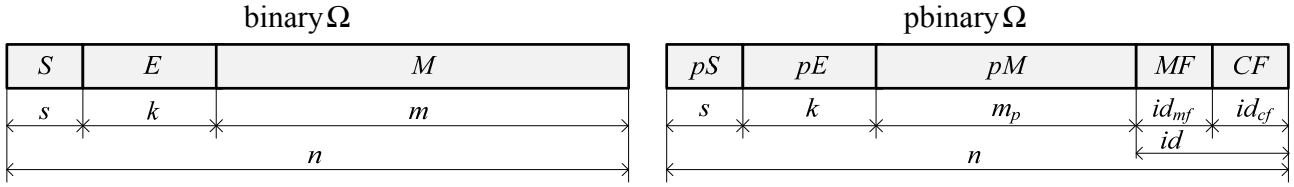


Рисунок 4.1 – Структура числа в стандартном формате binaryΩ и в эквивалентном ему формате pbinaryΩ с указанием полей формата: S и pS – поля знака, E и pE – поля порядка e со смещением, M и pM – поля остатка мантиссы μ', MF – поле модификатора формата, CF – поле кода формата ($n \equiv \Omega$ – количество разрядов формата числа)

В таблице 4.1 приведены значения длин полей модифицированного формата pbinaryΩ из рисунка 4.1 и значение поля CF.

В полученных форматах поле мантиссы числа модифицировано путем выделения необходимого количества бит для поля идентификатора формата длиной id, состоящего из поля модификатора MF длиной id_{mf} (определяет внутреннюю модификацию формата) и кода формата CF длиной id_{cf} (является признаком принадлежности к модифицированному формату).

Таблица 4.1 – Разрядность полей pbinaryΩ с указанием значения кода формата

Ω	*CF*	*s*	*k*	m_p	*id*	id_{mf}	id_{cf}
32	0	1	8	21	2	1	1
64	01	1	11	48	4	2	2
128	011	1	15	104	8	5	3
256	0111	1	20	219	16	12	4

Формирование кода идентификатора *CF* определено таким образом (см. таблицу 4.1), что по положению младшего нуля относительно указателя *pnt* на начальный (младший) разряд поля формата числа можно точно выделить из памяти все разряды формата pbinaryΩ, принадлежащие диапазону [*pnt* + *n* –1: *pnt*].

В зависимости от модификатора *MF*, формат pbinaryΩ может содержать в себе другие подформаты представления данных. Для таких форматов примем следующий синтаксис (в квадратные скобки взяты необязательные обозначения):

pbinaryΩ[/Ψ][ϕ],

где Ψ – точность формата, в 2 (для $\phi \in \{f, i, p\}$) или 4 (для $\phi \in \{fp, ip\}$) раза меньшая Ω и показывающая формат составной части чисел, заключенных в определяемую разрядность Ω; ϕ – указатель типа числа с плавающей запятой (f – дробное двоичное число, i – интервальное двоичное число, p – тетракод) или типа постбинарного числа (fp – тетракод дробного числа и ip – тетракод интервального числа).

В таблицах 4.2 и 4.3 приведены все модификации форматов чисел различной точности в зависимости от значения поля *MF*.

Таблица 4.2 –Предлагаемые модификации форматов pbinaryΩ
(при Ω = 32 и Ω = 64)

Модификатор MF_n[1:0]	pbinary32 (*использует MF*[0])	pbinary64 (*использует MF*[1:0])
00	pbinary32	pbinary64
01	pbinary32/16p	pbinary64/32f
10		pbinary64/32i
11		pbinary64/32p

Таблица 4.3 – Предлагаемые модификации форматов pbinaryΩ
(при Ω = 128 и Ω = 256)

Модификатор MF_n [11:0]	pbinary128 (*использует MF*[4:0])	pbinary256 (*использует MF*[11:0])
0000 0000 0000	pbinary128	pbinary256
0000 0000 0001	pbinary128/64f	pbinary256/128f
0000 0000 0010	pbinary128/64i	pbinary256/128i
0000 0000 0011	pbinary128/64p	pbinary256/128p
0000 0000 0100	pbinary128/32fp	pbinary256/64fp
0000 0000 0101	pbinary128/32ip	pbinary256/64ip
…	резерв	резерв

4.2 Структура, назначение и основные определения модифицированных форматов чисел с плавающей запятой

Рассмотрим структуру, особенности формирования и назначение каждой из представленных модификаций pbinary (в качестве примера на рисунке 4.2 представлены все модификации формата pbinary128).

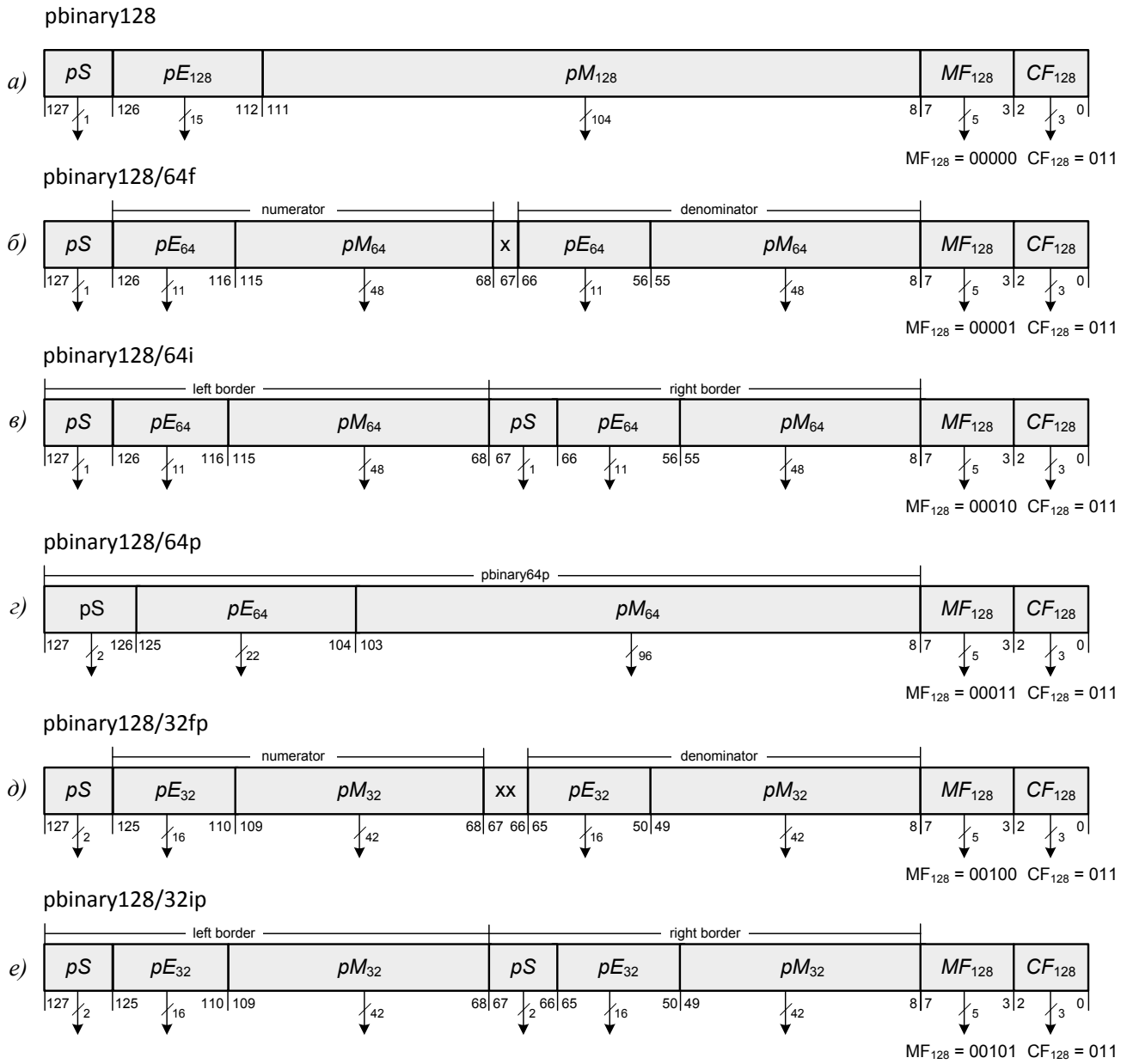


Рисунок 4.2 – Возможные модификации форматов чисел с разрядностью $\Omega = 128$ (numerator – числитель дроби; denominator – знаменатель дроби; left border – левая граница интервала; right border – правая граница интервала): *а* – двоичный числовой формат; *б* – двоичный дробный формат; *в* – двоичный интервальный формат; *г* – постбинарный числовой формат; *д* – постбинарный дробный формат; *е* – постбинарный интервальный формат

Определение 3. Модифицированный формат pbinaryΩ (или сокращенно pbΩ) при $\Omega \in \{32, 64, 128, 256\}$ (см. рисунок 4.2,*а*) – это формат чисел с плавающей запятой, который по организации полей и принципам кодирования

полностью соответствует стандартному формату binaryΩ, однако имеет модифицированное поле *pM* по отношению к *M*, длина которого уменьшена на *id* разрядов. Для формата pbinaryΩ справедливы следующие соотношения, связанные с определением количества разрядов полей (Рисунок 4.1):

1) Значение *id* – количество разрядов идентификатора формата:

$$id = \frac{\Omega}{16}; \tag{4.4}$$

2) Значение id_{cf} – разрядность кода формата:

$$id_{cf} = \log_2 id = \log_2 \frac{\Omega}{16};$$

3) Значение id_{mf} – разрядность модификатора формата:

$$id_{mf} = id - id_{cf} = \frac{\Omega}{16} - \log_2 \frac{\Omega}{16};$$

4) Значение *m* – количество разрядов мантиссы:

$$m_p = m - id. \tag{4.5}$$

Согласно определению 3, организация полей знака *pS* и порядка *pE* остаются без изменения и полностью эквивалентны соответствующим полям *S* и *E*. В таблице 4.4 показаны разрядности форматов pbinaryΩ по отношению к binaryΩ.

Каждое число, представленное в формате pbinaryΩ, имеет знаковый бит *pS* (Рисунок 4.1): $pS = 0$ для положительного и $pS = 1$ для отрицательного числа. Значение смещенной экспоненты *n*-разрядного формата с *k*-разрядным полем порядка формируется следующим образом:

$$pE_{n,k} = e + offset_k, \tag{4.6}$$

где *e* – порядок числа, определенный в выражениях (4.1–4.3); $offset_k$ – заданное смещение для порядка длиной *k* разрядов в формате pbinary:

$$offset_k = 2^{(k-1)} - 1. \tag{4.7}$$

В поле *pM* записывается остаток мантиссы двоичного нормализованного числа с плавающей точкой – отброшена старшая 1 от нормализованного двоичного числа вида (4.3).

Таблица 4.4 – Разрядность полей стандартных и модифицированных форматов чисел (подчеркиванием выделены форматы стандарта IEEE 754-2008)

Формат	*Количество двоичных разрядов полей*				*Точность*	*Смещение порядка: offset*
	s	*k*	*m* (m_p)	*n*		
binary16	1	5	10	16	Half (половинная точность)	+15
binary32	1	8	23	32	Single (одинарная точность)	+127
pbinary32	1	8	(21)			
binary64	1	11	52	64	Double (двойная точность)	+1023
pbinary64	1	11	(48)			
binary128	1	15	112	128	Quadruple (квадро-точность	+16 383
pbinary128	1	15	(104)			
pbinary256	1	20	(219)	256	Octuple (октоточность)	+524 287

Так как значения всех полей формата pbinaryΩ записываются в двоичном виде, для pbinaryΩ справедливы следующие соотношения [68, с. 95]:

1) Определение значения двоичного нормализованного числа из полей формата (все множители – в 2 с/с):

$$d_{m_p,k}^{norm} = (-1)^S \times 1, pM_{m_p} \times 2^{(pE_k - offset_k)}. \quad (4.8)$$

2) Определение значения десятичного нормализованного числа из полей формата (все множители – в 10 с/с):

$$x_{m_p,k}^{norm} = (-1)^S \cdot 2^{pE_k - offset_k} \cdot \left(1 + \frac{pM_{m_p}}{2^{m_p}}\right). \quad (4.9)$$

Аналогично binaryΩ, формат pbinaryΩ имеет ряд исключительных чисел, к которым нельзя применять формулы (4.30–4.33):

1. Положительный и отрицательный нули: +0 (pS = 0; pE = 0; pM = 0) и –0 (pS = 1; pE = 0; pM = 0).

2. Положительная и отрицательная бесконечности: +*infinity* ($pS = 0$; $pE = 111\ldots11$; $pM = 0$) и –*infinity* ($pS = 1$; $pE = 11\ldots1$; $pM = 0$). Эти числа сигнализируют о превышении границы диапазона представления чисел в текущем формате.

3. Не числа – NaN (No a Numbers), к которым относятся символы, или результаты недопустимых операций. При pE равному 111…11 и pM – любому ненулевому значению, различают +NaN ($pS = 0$) и –NaN ($pS = 1$).

4. Денормализованные числа – числа, мантиссы которых лежат в диапазоне $0{,}1 \le pM < 1$. Признак денормализованного числа в формате – все разряды порядка pE равны 0. Денормализованные числа находятся ближе к нулю, чем нормализованные, и разбивают минимальный разряд нормализованного числа на некоторое подмножество. Для ненормализованных чисел справедливы следующие определения [68, с. 148]:

1) Определение значения двоичного денормализованного числа из полей формата (все множители должны быть представлены в 2 с/с):

$$d_{m_p,k}^{dnrm} = (-1)^S \times 0, pM_{m_p} \times 2^{-offset_k}. \tag{4.10}$$

2) Определение значения десятичного денормализованного числа из полей формата (все множители – в 10 с/с):

$$x_{m_p,k}^{dnrm} = (-1)^S \cdot 2^{(1-offset_k)} \cdot \frac{pM_{m_p}}{2^{m_p}}. \tag{4.11}$$

На рисунке 4.3 представлено отображение чисел x_i форматов binaryΩ ($i = 1$) и pbinaryΩ ($i = 2$) на числовой оси.

Совокупности областей ±Norm и ±Denorm являются диапазоном представления положительных или отрицательных чисел формата pbinaryΩ на всей числовой оси. Используя соотношения (4.9) и (4.11), получим десятичные значения модулей минимальной и максимальной границ диапазонов представления чисел в формате pbinaryΩ, которые приведены в таблицах 4.5 и 4.6.

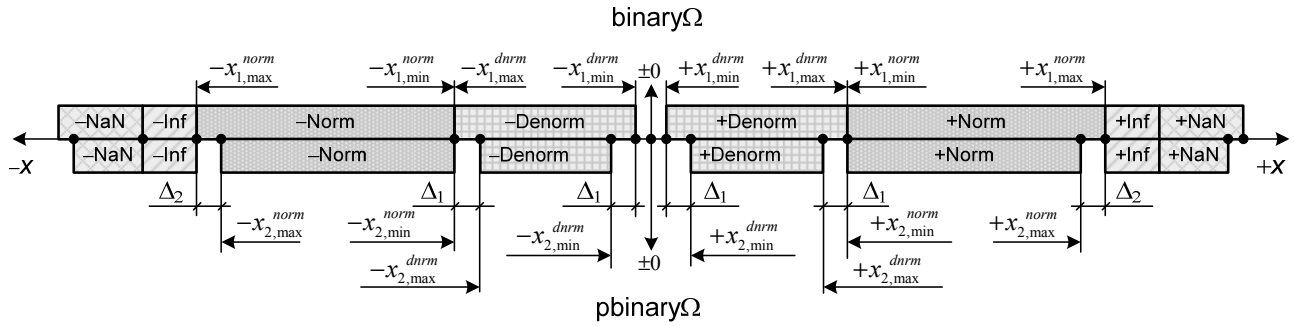


Рисунок 4.3 – Отображение чисел форматов binaryΩ и pbinaryΩ на числовой оси (±Norm, ±Denorm – области нормализованных и ненормализованных чисел в положительной и отрицательной полуосях)

Уменьшенная при модификации разрядность мантиссы постбинарного формата приводит к сужению диапазона представления денормализованных (близких к нулю) чисел по отношению к эквивалентному двоичному формату binaryΩ на значения Δ_1, а также уменьшение диапазона нормализованных чисел со стороны максимально представимых чисел на значение Δ_2. Сопоставляя формулы (4.9) и (4.11), получим значения Δ_1 и Δ_2 погрешности границ диапазона представления чисел по отношению к формату binaryΩ:

$$\Delta_1 = \left| x_{1,\min}^{dnrm} - x_{2,\min}^{dnrm} \right| = \left| x_{1,\max}^{dnrm} - x_{2,\max}^{dnrm} \right| = 2^{(1-offset-m_p)} \cdot \left(1 - 2^{-id}\right); \quad (4.12)$$

$$\Delta_2 = \left| x_{1,\max}^{norm} - x_{2,\max}^{norm} \right| = 2^{(offset-m_p)} \cdot \left(1 - 2^{-id}\right). \quad (4.13)$$

Здесь *id* – количество разрядов идентификатора формата, m_p – длина мантиссы и *offset* – смещение порядка модифицированного формата binaryΩ (Рисунок 4.1).

Рассмотрим полученные диапазоны чисел из рисунке 4.3 модифицированных и стандартных форматов. Получаем

$$\left[| x_{2,\min}^{dnrm} |,\ | x_{2,\max}^{dnrm} | \right] = \left[| x_{1,\min}^{dnrm} | + \Delta_1,\ | x_{1,\max}^{dnrm} | - \Delta_1 \right] =$$

$$= \left[2^{id} \cdot | x_{1,\min}^{dnrm} |,\ 2^{-id} \cdot | x_{1,\max}^{dnrm} | \right]. \quad (4.14)$$

Таблица 4.5 – Значения границ диапазона ненормализованных чисел (±Denorm) модифицированного формата pbinaryΩ

Ω	$\left\|x_{2,\min}^{dnrm}\right\|$	$\left\|x_{2,\max}^{dnrm}\right\|$	Δ_1
32	$2^{-147} \approx 5,60519386 \cdot 10^{-45}$	$2^{-126} \cdot (1-2^{-21}) \approx$ $\approx 1,17549379 \cdot 10^{-38}$	$\approx 4,204 \cdot 10^{-45}$
64	$2^{-1070} \approx 7,90505033 \cdot 10^{-323}$	$2^{-1022} \cdot (1-2^{-48}) \approx$ $\approx 2,22507386 \cdot 10^{-308}$	$\approx 7,411 \cdot 10^{-323}$
128	$2^{-16486} \approx 1,65764483 \cdot 10^{-4963}$	$2^{-16382} \cdot (1-2^{-104}) \approx$ $\approx 3,36210314 \cdot 10^{-4932}$	$\approx 1,651 \cdot e^{-4963}$
256	$2^{-524505} \approx 1,8286336 \cdot 10^{-157892}$	$2^{-524286} \cdot (1-2^{-219}) \approx$ $\approx 1,54061213 \cdot 10^{-157826}$	$\approx 1,829 \cdot 10^{-157892}$

Таблица 4.6 – Значения границ диапазона нормализованных чисел (±Norm) модифицированного формата pbinaryΩ

Ω	$\left\|x_{2,\min}^{norm}\right\|$	$\left\|x_{2,\max}^{norm}\right\|$	Δ_2
32	$2^{-126} \approx 1,17549379 \cdot 10^{-38}$	$2^{127} \cdot (2-2^{-21}) \approx$ $\approx 3,40282286 \cdot 10^{38}$	$\approx 6,085 \cdot 10^{31}$
64	$2^{-1022} \approx 2,22507386 \cdot 10^{-308}$	$2^{1023} \cdot (2-2^{-48}) \approx$ $\approx 1,79769313 \cdot 10^{308}$	$\approx 2,994 \cdot 10^{293}$
128	$2^{-16382} \approx 3,36210314 \cdot 10^{-4932}$	$2^{16383} \cdot (2-2^{-104}) \approx$ $\approx 1,18973150 \cdot 10^{4932}$	$\approx 2,921 \cdot 10^{4900}$
256	$2^{-524286} \approx 1,54061213 \cdot 10^{-157826}$	$2^{524287} \cdot (2-2^{-219}) \approx$ $\approx 2,5963757 \cdot 10^{157826}$	$\approx 1,541 \cdot 10^{157760}$

Равенство (4.14) позволяет сформулировать *следующие утверждения*:

1. Модуль минимального денормализованного числа формата pbinaryΩ больше модуля аналогичного числа стандартного формата binaryΩ в 2^{id}

раза, а модуль максимального ненормализованного числа формата pbinaryΩ меньше модуля аналогичного числа формата binaryΩ в 2^{id} раза.

2. Диапазон денормализованных чисел ±Denorm формата pbinaryΩ сужен относительно соответствующего диапазона формата binaryΩ в 2^{id+1} раза.

По отношению к диапазону нормализованных чисел модифицированного и стандартного форматов с плавающей запятой, сформулируем и докажем следующую лемму:

Лемма 2. Отношение максимальных границ диапазона нормализованных чисел с плавающей запятой для стандартного (binaryΩ) и модифицированного (pbinaryΩ) форматов одной и той же точности Ω/32 близко к единице.

Доказательство: Десятичное значение модуля минимального нормализованного числа $|x_{1,\min}^{norm}|$ соответствует аналогичному числу $|x_{2,\min}^{norm}|$, поскольку в формировании этих чисел участвуют только значения полей знака и порядка, которые идентичны в соответствующих постбинарных и бинарных форматах (рисунок 4.3). Таким образом,

$$\left[|x_{2,\min}^{norm}|,\ |x_{2,\max}^{norm}|\right]=\left[|x_{1,\min}^{norm}|,\ |x_{1,\max}^{norm}|-\Delta_2\right]=$$

$$=\left[|x_{1,\min}^{norm}|,\ \frac{2^{-(m_p+1)-id}-1}{2^{-(m_p+1)}-1}\cdot|x_{1,\max}^{norm}|\right]. \qquad (4.15)$$

Рассмотрим функцию $f(a,b)=\dfrac{2^{-(a+1)-b}-1}{2^{-(a+1)}-1}$, полученную в (4.15). Имеем предел функции, равный единице: $\lim\limits_{a\to\infty} f(a,b)=1$, откуда следует, что $f(a,b)\approx 1$ при возрастающем аргументе a. Соответственно, для значений m_p и id представленных форматов справедливо соотношение

$$\frac{2^{-(m_p+1)-id}-1}{2^{-(m_p+1)}-1}\approx 1, \qquad (4.16)$$

причем данное приближенное равенство стремится к строгому при возрастании значения m_p, что и *требовалось доказать*.

Учитывая лемму 2, окончательно получаем:

$$\left[| x_{2,\min}^{norm} |, | x_{2,\max}^{norm} | \right] \approx \left[| x_{1,\min}^{norm} |, | x_{1,\max}^{norm} | \right]. \quad (4.17)$$

Согласно выражениям (4.15) и (4.17) становится очевидным, что с позиции повышения эффективности вычислений, использование модифицированных форматов предпочтительней в области нормализованных чисел, чем реализация алгоритмов вычислений с денормализованными числами. Кроме того, для обработки денормализованных чисел применяются собственные (отличные от работы с нормализованными числами) алгоритмы вычислений, что сказывается на быстродействии и на эффективности компьютерной арифметики в целом.

Поэтому в модифицированных форматах работу с денормализованными числами можно исключить, руководствуясь следующей леммой:

Лемма 3. Множество всех денормализованных чисел формата исходной точности полностью принадлежит множеству нормализованных чисел формата, имеющего точность, большую исходной.

Доказательство данной леммы следует из сопоставления результирующих чисел, полученных из формул (4.7), (4.9) и (4.11) при различных значениях m_p, k и $offset_k$, соответствующих форматам «соседних» по классу точности, а также при анализе граничных значений из таблиц 4.6 и 4.7. Например, предположим, что модуль минимального денормализованного числа $\left| x_{m_p,k}^{dnrm} \right|$ постбинарного формата одинарной точности pbinary32 ($m_p = 23,\ k = 8,\ pM_{m_p} = 1$) находится в области нормализованных чисел аналогичного формата двойной точности pbinary64, т. е. больше его модуля минимального нормализованного числа $\left| x_{m_p,k}^{norm} \right|$ ($m_p = 48$, $k = 11$, $pM_{m_p} = 0$, $pE_k = 1$). Тогда доказательством данной леммы будет являться справедливость следующего неравенства:

$$\left(\left| x_{23,\ 8}^{dnrm} \right| = 2^{(1-2^{(7-1)}+1)} \cdot \frac{1}{2^{23}} \right) > \left(2^{(1-2^{(11-1)}+1)} \cdot \left(1 + \frac{0}{2^{48}} \right) = \left| x_{48,\ 11}^{norm} \right| \right),$$

откуда $\left(2^{-62} \cdot \frac{1}{2^{23}} \right) > 2^{-1022}$, и окончательно получаем $\frac{1}{2^{85}} > \frac{1}{2^{1022}}$. Неравенство справедливо, что и *требовалось доказать*.

Таким образом, согласно лемме 3, всякое денормализованное число формата pbinary32 можно заменить нормализованным числом формата pbinary64, денормализованное число формата pbinary64 – нормализованным числом формата pbinary128, и т. д. Всякое денормализованное число модифицированного формата старшего класса точности (в нашем случае pbinary256 – формат восьмерной точности) можно представить как нулевое значение, поскольку абсолютная погрешность представления такого числа, согласно таблице 4.6, не будет превосходить $1{,}54 \cdot 10^{-157826}$.

Сформулируем определения составных модифицированных форматов.

Определение 4. Модифицированный формат pbinary**Ω/Ψ**f («дробный» формат, сокращенная запись: pb**Ω/Ψ**f), см. рисунок 4.2 *б* – формат чисел с плавающей запятой по способу кодирования и организации полей полностью соответствующий модифицированному формату pbinary**Ψ** (при этом **Ψ** = **Ω**/2), и содержит два числа точности **Ψ**/32: числитель (поля мантиссы порядка и знака всей дроби в старших разрядах) и знаменатель (поля мантиссы и порядка в младших разрядах) дроби вида a/b, где $a,b \in \mathbb{R}$.

На основании определения 4, использование формата pb**Ω/Ψ**f для чисел x_1 и x_2, представленных в стандартном формате binary**Ψ**, над которыми нужно выполнить операцию деления, обусловлено сложностью общего цикла вычислений, и возможностью отложить деление, т. е. сохранить результат в виде дроби x_1/x_2 до заключительного этапа вычислений.

Определение 5. Модифицированный формат pbinary**Ω/Ψ**i («интервальный» формат, сокращенная запись: pb**Ω/Ψ**i), см. рисунок 4.2 *в* – это формат чисел с плавающей запятой по способу кодирования и организации полей полностью соответствующий модифицированному формату pbinary**Ψ** (при этом **Ψ** = **Ω**/2), и содержит два числа точности **Ψ**/32: левую (поля мантиссы порядка и знака в старших разрядах) и правую (поля мантиссы порядка и знака в младших разрядах) границы вещественного интервала.

Заполнение полей модифицированного «интервального» формата аналогично «дробному», за исключением наличия знаковых полей для каждого значения границ интервала. Числа в формате pbΩ/Ψi могут быть использованы для решения задач в рамках интервального анализа или интервальной математики, где они будут представлять собой интервальный тип данных, вычислительные операции с которыми исключат возможные ошибки округления [36].

Определение 6. Модифицированный формат pbinaryΩ/Ψp («постбинарный», сокращенная запись: pbΩ/Ψp), см. рисунок 4.3,*г* – это формат чисел с плавающей запятой по организации полей полностью соответствующий модифицированному формату pbinaryΨ (при этом $\Psi = \Omega/2$), в котором число точности Ψ/32 представлено тетракодом, тетриты которого записаны парами битов, а совокупность этих битов совместно с битами идентификатора длиной *id* двоичных разрядов составляют Ω-разрядное битовое поле формата.

При формировании разрядности знака и порядка и мантиссы «постбинарного» формата pbΩ/Ψp справедливы следующие соотношения (согласно рисунку 4.1): $s_{\Omega} = 2s_{\Psi}$, $k_{\Omega} = 2k_{\Psi}$, $m_{p,\,\Omega} = 2m_{p,\Psi}$.

Определение 7. Модифицированный формат pbinaryΩ/Ψfp («постбинарной дробный» формат, сокращенная запись: pbΩ/Ψfp), см. рисунок 4.2,*д* – это формат чисел с плавающей запятой по способу кодирования и организацией полей соответствующий объединению pbΩ/Ψf ∪ pbΩ/Ψp, в результате чего Ψ = Ω/4.

Определение 8. Модифицированный формат pbinaryΩ/Ψip («постбинарный интервальный», или сокращенно pbΩ/Ψip), (см. рисунок 4.2,*е*) – это формат чисел с плавающей запятой по способу кодирования и организацией полей соответствующий объединению pbΩ/Ψi ∪ pbΩ/Ψp, в результате чего Ψ = Ω/4.

На рисунке 4.4 приведены соотношения модифицированных форматов, согласно определениям 4.3–4.8.

Основные характеристики каждого из определенных выше модифицированный форматов приведены в приложении И.

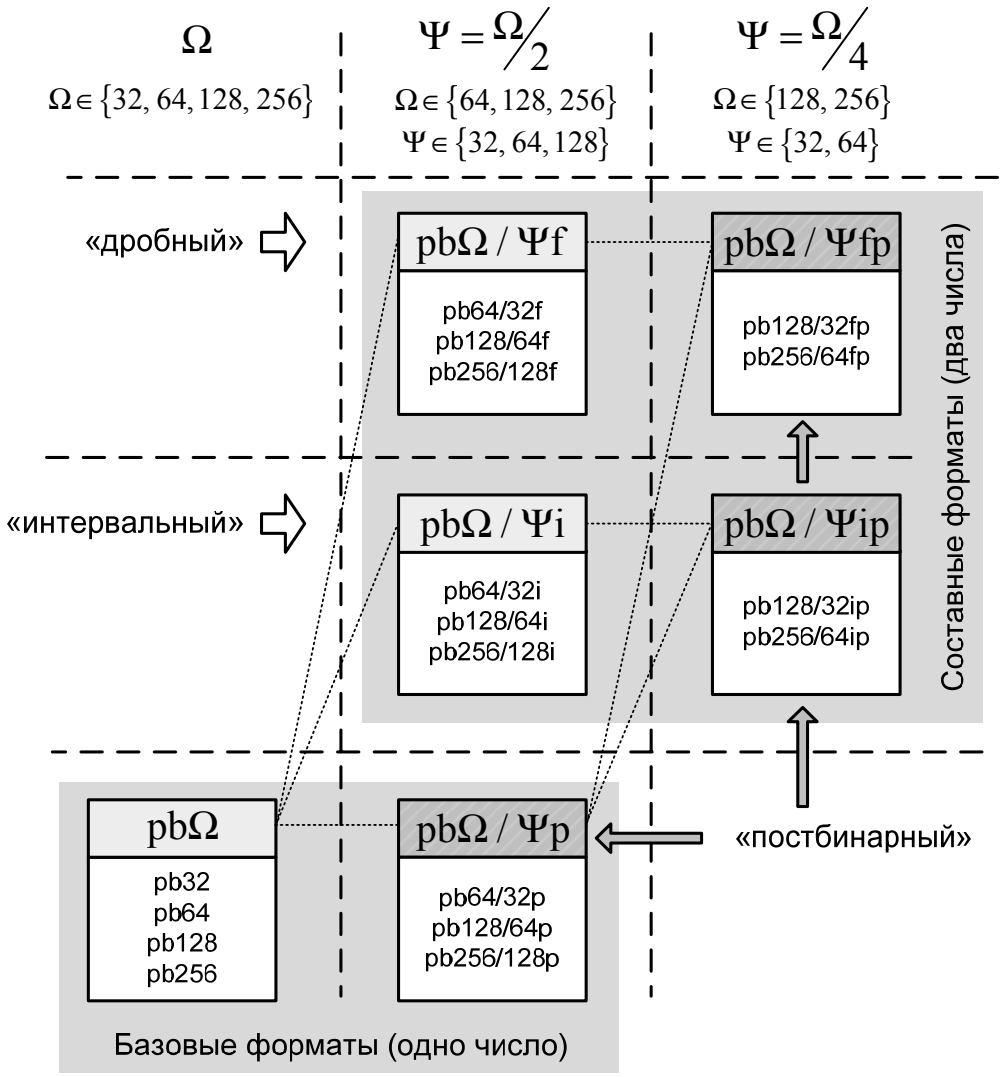


Рисунок 4.4 – Соотношение модифицированных форматов с указанием способов кодирования и назначения

4.3 Оценка погрешности представления чисел в модифицированных форматах чисел с плавающей запятой

При представлении числа в виде полей порядка, мантиссы и знака, на вещественной оси можно отложить конечный набор значений, в общем случае не превосходящий

$$K_\Omega = 2^\Omega = 2^{s+k+m_p}, \tag{4.18}$$

где K_Ω – количество чисел в модифицированном формате с плавающей запятой, представленное Ω-разрядным двоичным значением; s, k, m_p – разрядности знака, порядка и мантиссы соответственно.

Имея конечное количество кодируемых в формате с плавающей запятой значений (точек разрядной сетки) вещественной оси, вся дальнейшая процедура представления любого вещественного числа сводится к отображению его одной (чаще всего ближайшей) точкой разрядной сетки формата. Процедура подбора такой точки для исходного вещественного числа называется округлением числа, а расстояние между действительной позицией числа на вещественной оси и базовой точкой его отображения – абсолютной погрешностью Δ представления числа в формате с плавающей запятой.

Естественно, двоичные эквиваленты некоторых десятичных дробей совпадают с базовыми точками (при условии, что само число входит в диапазон представления данного формата с плавающей запятой). В этом случае такое число представляется в формате с плавающей точкой без округления (и, соответственно, без потери точности). К ним относятся числа, не имеющие дробную часть (множество целых чисел) и числа, которые можно представить в виде конечной двоичной дроби, дробная часть которых кратна 2^{-z}, где $z \in \mathbb{Z}$ («кратность отрицательной степени двойки»).

Остальные числа представлены в формате с плавающей запятой приближенно, т. е. с некоторой ошибкой точности. На рисунке 4.5 приведен график ошибки точности (абсолютной погрешности) представления числа в модифицированном формате (аналогичный форматам стандарта IEEE 754-2008) при увеличении порядка на 1 (т. е. от e до e+1). При этом точки n_i – точки разрядной сетки, сформированные значениями поля мантиссы формата с плавающей запятой.

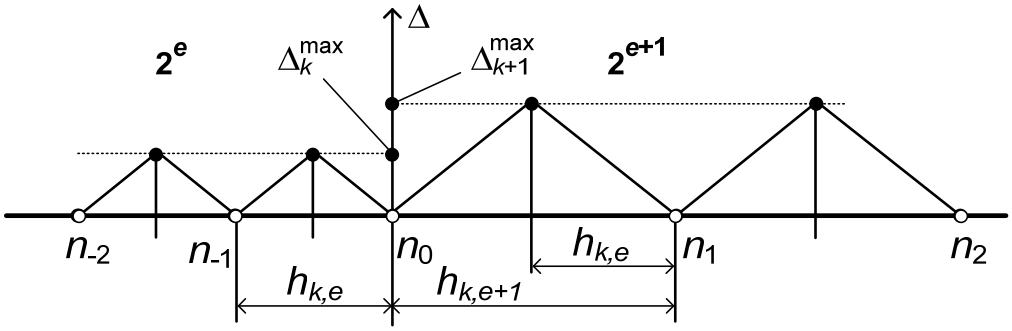


Рисунок 4.5 – График абсолютной погрешности представления чисел в форматах IEEE 754-2008 и модифицированных форматах

По форме графика видно, что максимальная абсолютная ошибка точности приходится на число, равноудаленное от двух подряд лежащих на вещественной оси базовых точек, а в самих базовых точках ошибка равна нулю (что говорит о точном представлении числа). Следовательно, чем больше расстояние между позицией исходного числа и ближайшей базовой точки, тем с меньшей точностью оно может быть представлено этой точкой. Расстояние $h_{к,e}$ будем называть *шагом чисел*, который является расстоянием между соседними точками разрядной сетки формата с порядком длиной k разрядов и значением e (согласно (4.1)), т. е. позициями чисел представленных полями форматов с единым десятичным значением порядка e ($e \neq 0$) и с различающимися на единицу мантиссами. Значение $h_{к,e}$ можно получить следующим образом:

$$h_{k,e} = 2^{e-m_p} = 2^{pE_k - offset_k - m_p}, \tag{4.19}$$

где $pE_k = e + offset_k, -$ десятичное число в k-разрядном поле порядка модифицированного формата с плавающей запятой (значение смещенного порядка); $offset_k$ – десятичное смещение порядка E_k; m – число разрядов поля мантиссы модифицированного формата.

Расстояние $h_{k,e}$ удваивается с увеличением порядка числа с плавающей запятой на единицу:

$$h_{k,e+1} = 2 \cdot h_{k,e}. \tag{4.20}$$

Иными словами, чем дальше от нуля, тем шире шаг чисел в формате IEEE 754–2008 по числовой оси.

Формулы (4.19) и (4.20) справедливы для расчета шага нормализованных чисел. Для денормализованных чисел (при $E_k = 0$ ($e = -offset_k$) и ненулевой мантиссе) шаг $h_{k,-offset_k}$ можно определить следующим образом:

$$h_{k,-offset_k} = 2^{1-m_p-offset_k}. \tag{4.21}$$

Очевидно, что $h_{k,-offset_k}$ – минимальный шаг, равный по своему значению модулю минимального денормализованного числа $\left|x_{1,\min}^{dnrm}\right|$ (см. рисунок 4.3).

Абсолютная максимальная ошибка для нормализованных ($\Delta_{k,e}^{\max}$) и денормализованных ($\Delta_{k,-offset_k}^{\max}$) чисел в модифицированном формате с плавающей запятой равна половине шага [147]:

$$\Delta_{k,e}^{\max} = \frac{1}{2}h_{k,e} = \frac{1}{2}\left(2^{e-m_p}\right) = 2^{pE_k-offset_k-m_p-1}; \tag{4.22}$$

$$\Delta_{k,-offset_k}^{\max} = \frac{1}{2}h_{k,-offset_k} = \frac{1}{2}\left(2^{1-offset_k-m_p}\right) = 2^{-(offset_k+m_p)}. \tag{4.23}$$

Учитывая (4.20) и (4.22), можно получить зависимость

$$\Delta_{k,e+1}^{\max} = 2 \cdot \Delta_{k,e}^{\max}, \tag{4.24}$$

т. е. *значение абсолютной максимальной ошибки точности представления удваивается с увеличением порядка числа с плавающей запятой на единицу*.

Относительная максимальная погрешность представления нормализованных ($\delta_{k,e}^{\max}$) и денормализованных ($\delta_{k,-offset_k}^{\max}$) чисел в модифицированных форматах равна

$$\delta_{k,e}^{\max} = \frac{\Delta_{k,e}^{\max}}{\left|x_{k,m_p}\right|} \cdot 100\%; \tag{4.25}$$

$$\delta_{k,-offset_k}^{\max} = \frac{\Delta_{k,-offset_k}^{\max}}{\left|x'_{k,m_p}\right|} \cdot 100\%, \tag{4.26}$$

где $\left|x_{k,m}\right|$ и $\left|x'_{k,m}\right|$ – модули десятичных чисел вида (4.2) и (4.3), полученные из модифицированных форматов представления нормализованных и денормализованных чисел с плавающей запятой соответственно.

Подставив (4.11) и (4.9) в (4.25) и (4.26), окончательно получаем

$$\delta_{k,e}^{\max} = \frac{1}{2^{m_p+1} + 2 \cdot pM} \cdot 100\%; \tag{4.27}$$

$$\delta_{k,-offset_k}^{\max} = \frac{1}{2 \cdot pM} \cdot 100\%, \tag{4.28}$$

Из (4.27) и (4.28) видно, что $\delta_{k,e}^{\max} << \delta_{k,-offset_k}^{\max}$, т. е. денормализованные числа представляются в форматах с плавающей запятой со значительно большей погрешностью, чем нормализованные. Этот факт подтверждает *справедливость использования только нормализованной формы числа* в модифицированных форматах.

Для стандартных форматов IEEE 754-2008 также справедливы формулы определения шага чисел, максимальной абсолютной и относительной погрешностей представления нормализованных и денормализованных чисел, если в выражениях (4.19)–(4.28) учитывать значения m вместо m_p и значения M вместо pM (согласно рисунку 4.1).

Формулы для нахождения значений максимальных абсолютной и относительной погрешностей представления нормализованных чисел для модифицированных форматов различной точности сведены в таблице 4.7.

Наличие ненулевых погрешностей представления чисел в модифицированных форматах с плавающей запятой является их недостатком, который компенсируется наличием форматов высокой точности, по сравнению со стандартными. Однако для получения достоверных результатов для компьютерных вычислений, использующих модифицированные форматы, необходима *модификация стандартных способов округления чисел*.

Таблица 4.7 – Формулы определения максимальной абсолютной и относительной погрешностей представления нормализованных чисел модифицированных форматов

Формат	k	$offset_k$	$\Delta_{k,e}^{\max}$	$\delta_{k,e}^{\max}$, %
pbinary32	8	127	2^{pE_k-149}	$(2^{22}+2\cdot pM)^{-1}\cdot 100\%$
pbinary64	11	1023	2^{pE_k-1072}	$(2^{49}+2\cdot pM)^{-1}\cdot 100\%$
pbinary128	15	16383	$2^{pE_k-16488}$	$(2^{105}+2\cdot pM)^{-1}\cdot 100\%$
pbinary256	21	524287	$2^{pE_k-524507}$	$(2^{220}+2\cdot pM)^{-1}\cdot 100\%$

4.4 Модификация стандартных способов округления чисел в форматах с плавающей запятой

Стандарт IEEE 754-2008 предусматривает четыре способа округления чисел, которые также справедливы для модифицированных форматов pbinary:

1. Округление к нулю (Рисунок 4.6 *а*). При округлении к нулю нужно просто отбросить незначащие разряды числа, поэтому этот способ самый легкий в аппаратной реализации.
2. Округление к ближайшему числу (к ближайшей точке разрядной сетки) (Рисунок 4.6 *б*). Данный способ округления в математическом сопроцессоре используется по умолчанию.
3. Округление к положительной бесконечности (Рисунок 4.6 *в*). Округление к $+\infty$ применяется при кодировании интервальных чисел [86].
4. Округление к отрицательной бесконечности (Рисунок 4.6 *г*), которое также применяется при кодировании интервальных чисел.

При этом для аппаратной поддержки стандартных видов округления достаточно проанализировать старший незначащий разряд двоичной дроби

исходного числа (т. е. первый разряд двоичного числа, не уместившийся в поле мантиссы), а также учитывать знак числа (Рисунок 4.7).

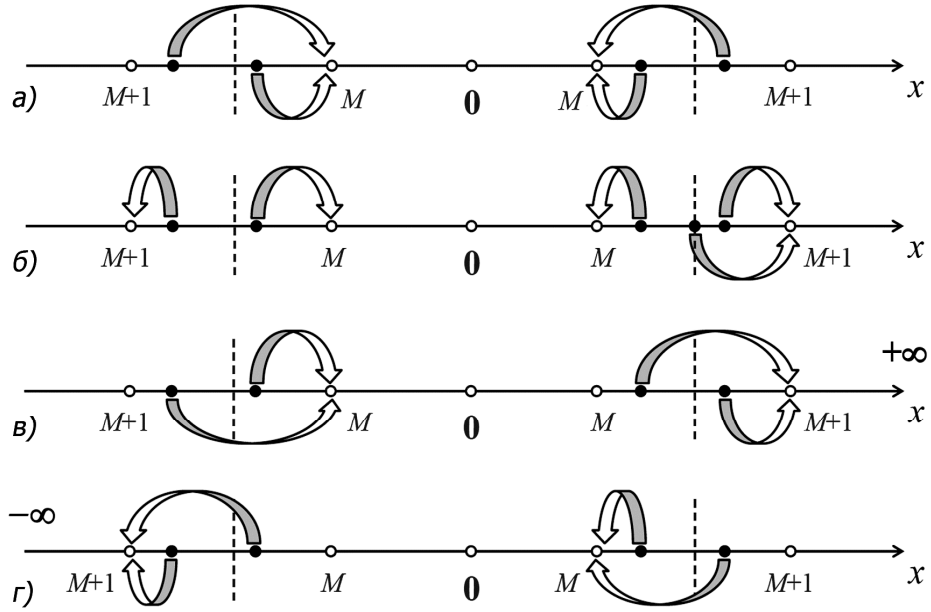


Рисунок 4.6 – Способы округления чисел стандарта IEEE 754-2008: *а* – к нулю; *б* – к ближайшему числу; *в* – к +∞; *г* – к –∞ (черные точки – действительные числа на числовой оси, белые точки – точки разрядной сетки формата с плавающей запятой)

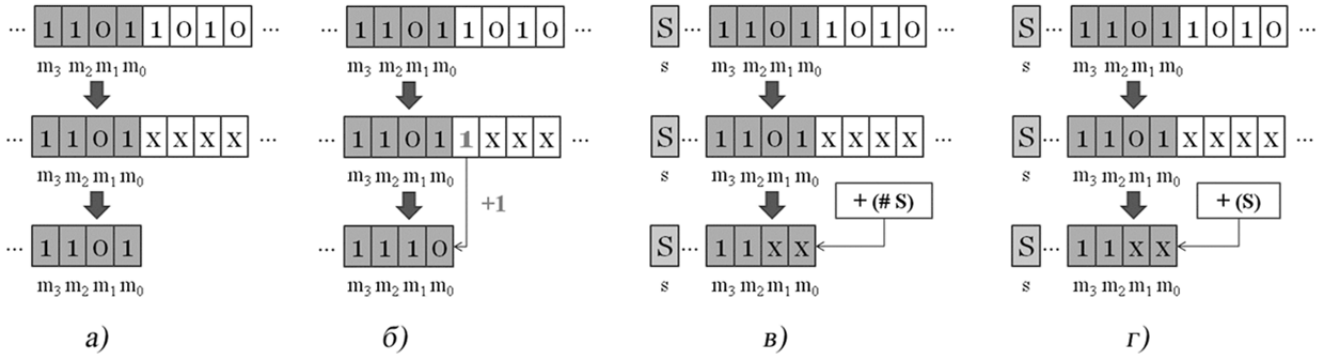


Рисунок 4.7 – Примеры реализации округления чисел стандарта IEEE 754-2008: *а* – к нулю; *б* – к ближайшему числу; *в* – к +∞; *г* – к –∞

Очевидно, что рассмотренные выше формулы абсолютной и относительной максимальной погрешности (4.22), (4.23), (4.27) и (4.28) справедливы только при выполнении округления к ближайшему числу.

На рисунке 4.8 приведена гистограмма с определенными значениями относительной погрешности для чисел в диапазоне от 536 870 784,0 до 536 871 168,0 с шагом 0,768 (500 измерений) при округлении к ближайшему числу.

Введение тетракода как системы кодирования в модифицированных «постбинарных» форматах pbΩ/Ψp, pbΩ/Ψfp и pbΩ/Ψip позволяет рассмотреть наряду со стандартными способами и новый, модифицированный способ округления.

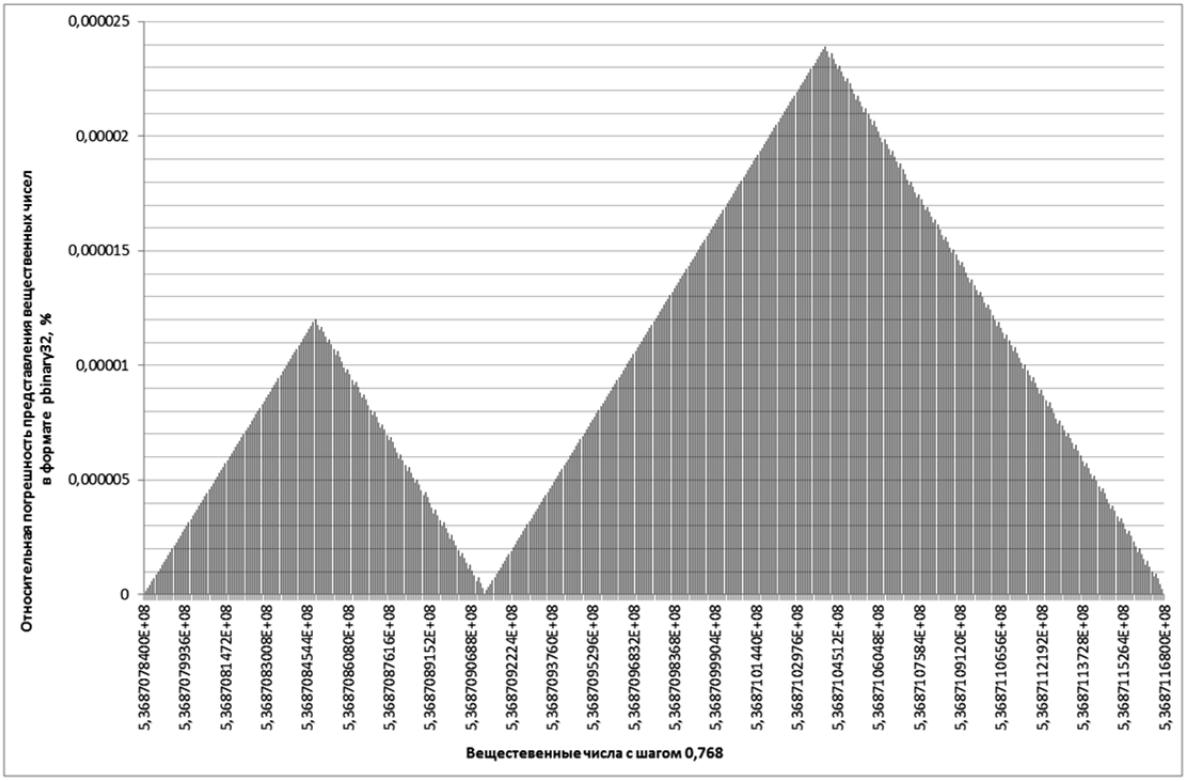


Рисунок 4.8 – Относительная погрешность вещественных чисел в формате pbinary32 при округлении к ближайшему числу

На рисунке 4.9 приведен способ модифицированного округления [148]. Вещественные числа, чьи позиции удалены от точек разрядной сетки формата (участки II и III), представляются в модифицированных форматах в виде интервального числа с границами соседних точек разрядной сетки [pM, pM+1].

Такая возможность достигается появлением значений M и A в младших разрядах мантиссы [80, с. 132–135].

Таким образом, исходя из рисунка 4.9, введем понятие постбинарного округления:

Определение 9. Постбинарное округление – модифицированный способ стандартного округления к ближайшему числу, в котором числа округляются в пределах ¼ шага числа между точками разрядной сетки формата числа с плавающей запятой.

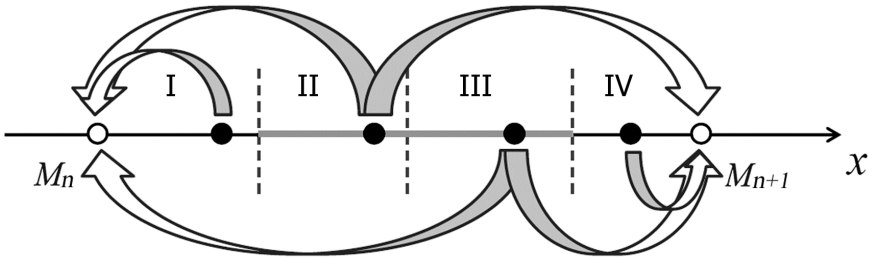


Рисунок 4.9 – Принцип модифицированного округления чисел в «постбинарных» форматах с плавающей запятой (черные точки – действительные числа на числовой оси, белые точки – точки разрядной сетки формата)

При постбинарном округлении рассматриваются значения двух старших незначащих разрядов (Рисунок 4.10):

1) при значениях, равных 00 – число находится в участке I вещественной оси – округление к меньшей точке разрядной сетки (т. е. к ближайшему числу): отбрасывание незначащих разрядов;
2) при значениях, равных 01 или 10 – число находится в участках II или III вещественной оси – происходит формирование нормированного тетракода [66, с. 241–244] в поле мантиссы путем прибавления к мантиссе значения M;
3) при значениях, равных 11 – число находится в участке IV вещественной оси – округление к большей точке разрядной сетки (т. е. к ближайшему числу): прибавление 1 к полю мантиссы.

Таким образом, при использовании постбинарного округления стало возможным:

1) для чисел, лежащих в областях I и IV вещественной оси, уменьшить ошибку точности представления числа, причем абсолютная максимальная погрешность $\Delta'^{\max}_{k,e}$ для этих чисел в постбинарном формате оказалась равной в пределе четверти шага чисел (Рисунок 4.11), что в два раза меньше аналогичной абсолютной ошибки $\Delta^{\max}_{k,e}$ для форматов стандарта IEEE 754-2008:

$$\Delta'^{\max}_{k,e} = \frac{1}{2}\Delta^{\max}_{k,e} = \frac{1}{4}h_{k,e} = 2^{pE_k - offset_k - m_p - 2}. \quad (4.29)$$

2) числа, лежащие в областях II и III вещественной оси, представлять интервалом, ширина которого равна шагу чисел: wid = $h_{k,e}$.

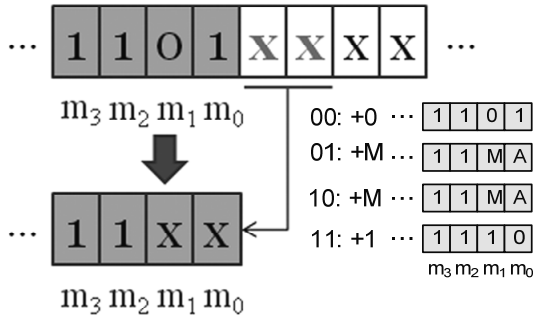


Рисунок 4.10 – Пример реализации постбинарного округления чисел для модифицированных «постбинарных» форматов с плавающей запятой

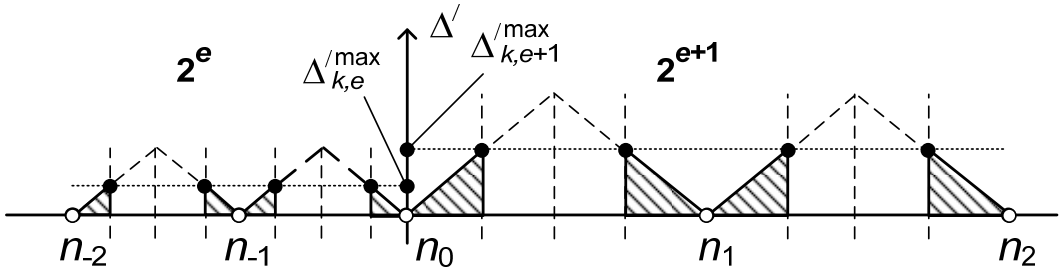


Рисунок 4.11 – График абсолютной погрешности представления чисел в модифицированных форматах при постбинарном округлении

На рисунке 4.12 приведена аналогичная рисунку 4.8 гистограмма с определенными значениями относительной погрешности при модифицированном округлении. Действительно, по отношению к рисунку 4.8, максимальные значения погрешностей уменьшились в 2 раза.

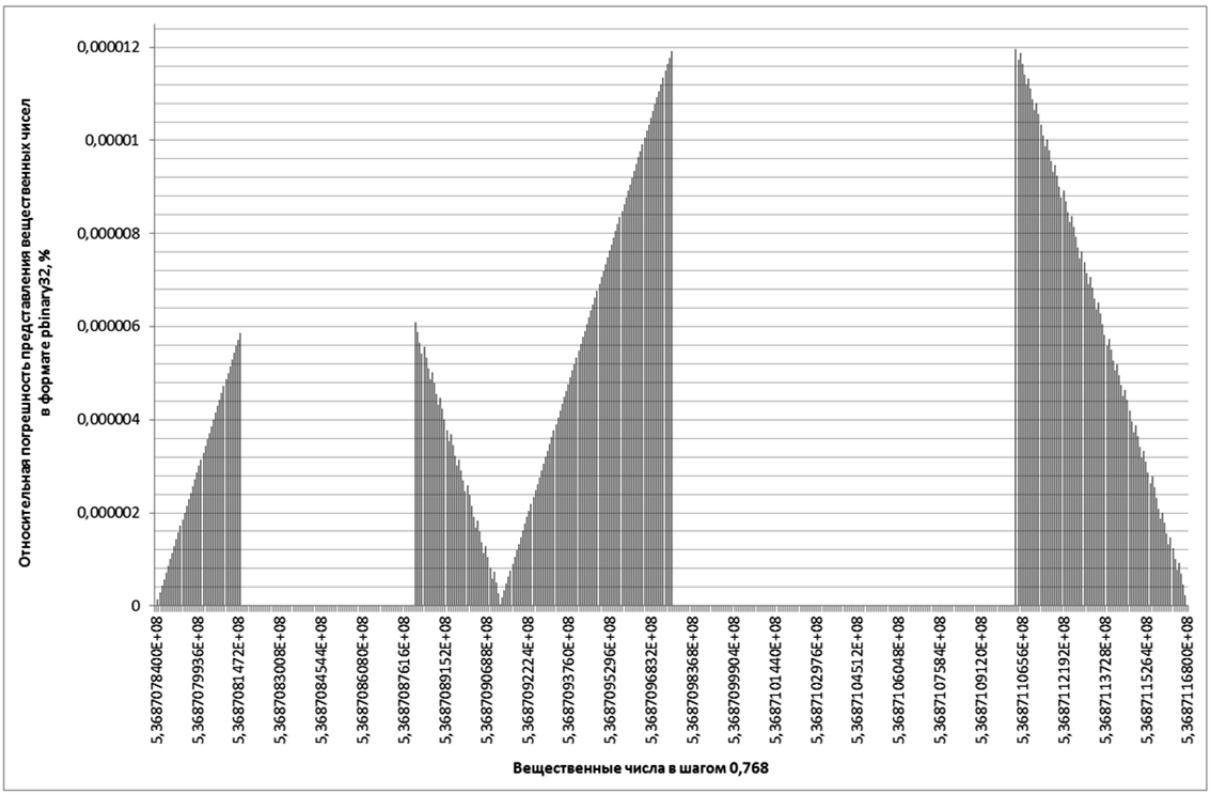


Рисунок 4.12 – Относительная погрешность вещественных чисел в формате pbinary32 при модифицированном округлении

Округление чисел с плавающей запятой очень важная проблема существующей плавающей арифметики, поскольку числа приходится округлять как при вводе исходных значений, так и после каждой арифметической операции. Переход к модифицированным «постбинарным» форматам частично исправляет данную ситуацию, так как в постбинарном представлении числа используется расширенный кодо-логический базис, позволяющий, использовать «гибкий» аппарат кодирования чисел, что в свою очередь обеспечивает:

- представление одним значением диапазона чисел на вещественной оси (например, в результате постбинарного округления);
- фиксирование и хранение незначащих (недоопределенных) разрядов мантиссы, приводящее к грамотному (с математической точки зрения) увеличению точности представления чисел;
- организацию плавающей арифметики с тетракодами, привносящей новые алгоритмы нормализации чисел, обработки исключительных ситуаций и базовых арифметических операций.

Следует отметить, при уменьшении в заложенной в формат числа (или величины) погрешности, увеличивается эффективность вычислений, поскольку повышается их достоверность. Это подтверждается тем, что при сложении и вычитании двух величин их абсолютные погрешности складываются; при умножении и делении двух величин на друга их относительные погрешности складываются; при возведении в степень приближенной величины ее относительная погрешность умножается на показатель степени [149].

4.5 Выводы к четвертой главе

Рассмотренные в данном разделе особенности модификации стандартных форматов чисел с плавающей запятой, модификация стандартного округления чисел, а также выведение формул для расчета погрешности представления чисел в этих форматах позволяют сделать следующие выводы:

1. Модификация форматов с плавающей запятой организована таким образом, что изменению подверглось только поле мантиссы, сохраняя при этом подобие модифицированных и стандартных форматов, что положительно сказывается на адаптации аппаратно-программного комплекса к обработке модифицированных форматов.

2. Наличие в модифицированных форматах поля идентификатора дает возможность компьютерным компонентам определять класс точности, тип и способ кодирования данных, находящихся в этих форматах.

3. Способность кодировать два действительных числа в одном поле модифицированного формата может быть полезна при реализации вычислений с интервальными форматами, а также для переноса операции деления к последнему шагу вычислений (реализация т. н. «отложенного деления»).

4. Возможность применения принципов тетракодирования, а, следовательно, и арифметики с тетракодами в «постбинарных» модифицированных форматах определяет переход от действительного представления числа к более расширенному – интервальному, что позволяет учитывать погрешности, возникшие в процессе вычислений.

5. Организация вычислений с модифицированными «постбинарными» форматами с плавающей запятой при использовании постбинарного округления позволяет повысить их надежность за счет уменьшения ошибки представления чисел, в частности, получение вдвое уменьшенной максимальной абсолютной погрешности для текущего шага чисел.

6. Реализация вычислительных алгоритмов на базе предложенных форматов позволяет обеспечить существенное расширение функциональных возможностей перспективных процессоров, в том числе за счет реализации текущего контроля требуемой разрядности и обеспечения на этой базе «гибкого форматирования» и «гибкой разрядности» при работе с компьютерным представлением численной информации.

ГЛАВА 5
ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ПОСТБИНАРНЫХ АРИФМЕТИКО-ЛОГИЧЕСКИХ СПОСОБОВ ОБРАБОТКИ ИНФОРМАЦИИ

5.1 Программная реализация постбинарного кодирования интервалов

На рисунке 5.1 представлены два варианта реализации программы [80, 87], которая позволяет декодировать 32-разрядный тетракод с учетом описанных в третьей главе (п. 3.1) преобразований.

Программа позволяет выполнить преобразование тетракодов в двоичные и десятичные наборы чисел. Исходный тетракод может быть задан как целочисленный (приводиться к двоичным 32-разрядным целым беззнаковым числам) либо вещественный» (приводиться к двоичным 32-разрядным числам с плавающей запятой).

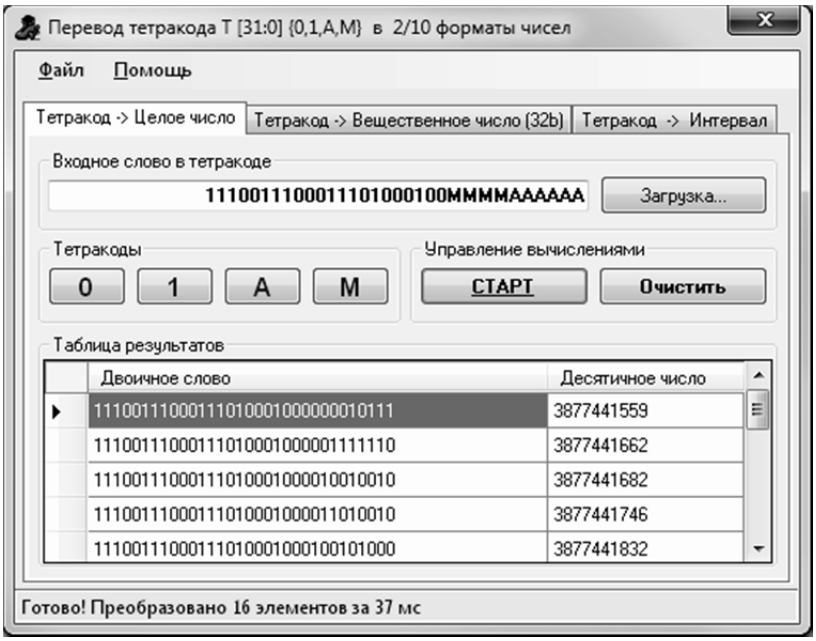


Рисунок 5.1 – Интерфейс программы «Перевод 32-разрядного тетракода в 2/10 форматы чисел» (результат работы режима «Тетракод → Целое число»)

Программа имеет 4 режима работы, в которых выполняются следующие преобразования:

1) Тетракод → Целое число: преобразование «целочисленного» тетракода в набор целых 32-разрядных чисел (Рисунок 5.1).

2) Тетракод → Целочисленный интервал: приведение тетракода к паре 32-разрядных целых чисел, являющихся границами интервала.

3) Тетракод → Вещественное число: преобразование тетракода в набор вещественных 32-разрядных чисел.

4) Тетракод → Вещественный интервал: приведение тетракода к паре 32-разрядных вещественных чисел, являющихся границами интервала.

В нижней части окна программы отображается сопроводительная информация, которая включает следующие параметры: время выполнения преобразования тетракода и количество полученных при этом элементов (режимы 1 и 3); середина, ширина и «собранные» границы интервала (режимы 2 и 4). Результирующий набор полученных значений представлен в двоичных и десятичных числах.

Рассмотрим декодирование тетракода на примере работы программы в 3-м и 4-м режимах. Значения тетритов исходного 32-разрядного тетракода *T*:

Знак		
0	11110011	0000000011100MMMMMMMMAAA
	Порядок	*Мантисса*

Результаты работы программы в режиме 3 (приведение тетракода к набору вещественных чисел) представлены на рисунке 5.2.

Таблица результатов

	Двоичное слово	Вещественное число
	0 11110011 00000000111000000000001	8,336071E+34
	0 11110011 00000000111000000001001	8,336079E+34
	0 11110011 00000000111000000010010	8,336088E+34
	0 11110011 00000000111000000011111	8,336101E+34
	0 11110011 00000000111000000100101	8,336107E+34

Готово! Преобразовано 128 элементов за 16 мс

Рисунок 5.2 – Результат сведения тетракода к набору вещественных чисел

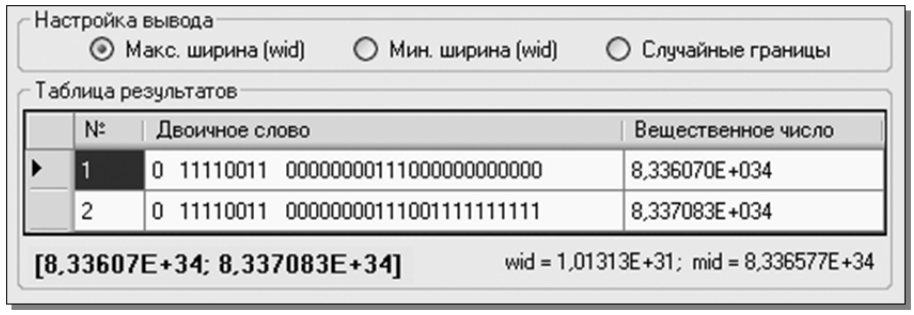
За 16 мс программа выполнила приведение тетракода T к набору из $2^8 = 128$ вещественных чисел (в тетракоде 8 тетритов M). При этом тетриты M в своей совокупности обеспечили полный перебор двоичных значений, а тетриты A в каждом числе из набора случайным образом свелись к нулевым или единичным битам. Повторные запуски программы приведут к получению различных по содержанию наборов чисел, поскольку тетриты A каждый раз будут заполняться случайными двоичными значениями.

В 4-м режиме работы программы (приведение тетракода к вещественному интервалу) возможны модификации получения результата: получение диапазонов колебаний границ («минимальная ширина интервала», «максимальная ширина интервала») и получение интервала с учетом случайного замещения двоичными значениями тетритов A. При отсутствии тетритов A в тетракоде, программа будет возвращать одинаковые интервалы не зависимо от выбранной модификации режима. На рисунке 5.3 представлены результаты работы программы при сведении тетракода T к интервальным границам в каждой модификации. Модификацию «максимальная ширина» (Рисунок 5.3 *а*) можно трактовать следующим образом: определение минимального и максимального значения в декодированном из тетракода наборе чисел.

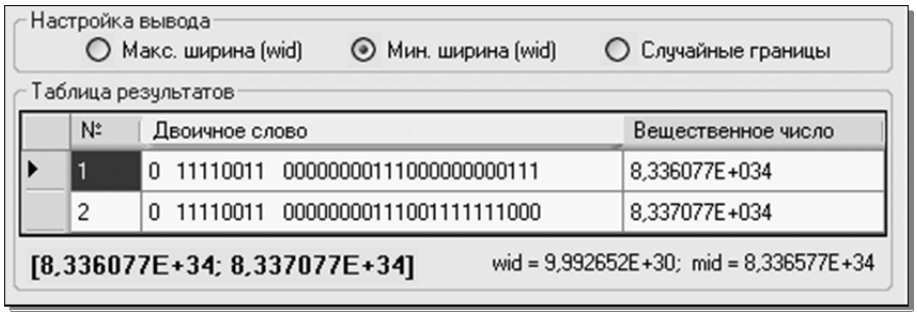
Действительно, каков бы ни был набор чисел (с учетом разрядов A), ни одно число не будет меньше левой и больше правой границ интервала «максимальной ширины», численные значения которых можно получить из тетракода T, используя возвращаемые значения тетрафункций MIN_A (MAX_A) – минимизации (максимизации) неопределенности и MIN_M (MAX_M) – минимизации (максимизации) множественности (Таблица 5.1) для левой и правой границы соответственно, т. е. [MIN_M(MIN_A(T)); MAX_M(MAX_A(T))] – искомый интервал.

Применение данной модификации представляет собой интервальную оценку «сверху», т. е. позволяет оценить максимальный «захват» участка числовой оси приведенным из тетракода интервалом.

а)



б)



в)

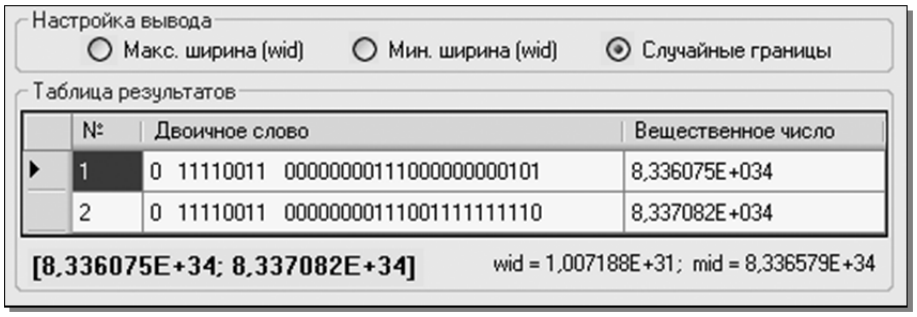


Рисунок 5.3 – Результат сведения тетракода к вещественному интервалу (*а* – получение интервала максимальной ширины; *б* – получение интервала минимальной ширины; *в* – получение случайных границ)

Модификация «минимальная ширина» (Рисунок 5.3 *б*) отображает «худшие» случаи относительно тетритов A при определении границ интервала. Границы интервала с минимальной шириной из тетракода T можно получить, используя следующие соотношения для левой и правой границы соответственно:

[MIN_M(MAX_A(T)); MAX_M(MIN_A(T))] – искомый интервал.

Применение данной модификации представляет собой интервальную оценку «снизу», т. е. позволяет оценить минимальный «захват» участка числовой оси приведенным из тетракода интервалом.

Таблица 5.1 – Функции минимизации и максимизации для тетритов A и M

Унарная функция	Значение тетрита *t*			
	0	1	A	M
MIN_M(t)	0	1	A	0
MAX_M(t)	0	1	A	1
MIN_A(t)	0	1	0	M
MAX_A(t)	0	1	1	M

Действительно, границы интервала (с учетом разрядов A), полученные при модификации «случайные границы» (Рисунок 5.3 *в*), никогда не будут меньше по модулю соответствующих границ интервала с минимальной шириной. Таким образом, границы сведенного из тетракода T интервала при любом случайном наборе бит для разрядов A будут принадлежать следующим диапазонам:

$[\mathrm{MIN\_M}(\mathrm{MIN\_A}(T));\ \mathrm{MIN\_M}(\mathrm{MAX\_A}(T))]$ – для левой границы;

$[\mathrm{MAX\_M}(\mathrm{MIN\_A}(T));\ \mathrm{MAX\_M}(\mathrm{MAX\_A}(T))]$ – для правой границы.

Ширина этих диапазонов определяет пределы колебаний границ интервала, полученного при декодировании тетракода T. Такой интервал может быть записан в виде

$$\begin{bmatrix} [\mathrm{MIN\_M}(\mathrm{MIN\_A}(T));\ \mathrm{MIN\_M}(\mathrm{MAX\_A}(T))]; \\ [\mathrm{MAX\_M}(\mathrm{MIN\_A}(T));\ \mathrm{MAX\_M}(\mathrm{MAX\_A}(T))] \end{bmatrix} \Rightarrow$$

$$\Rightarrow [\mathrm{MIN\_M}(\mathrm{Rnd\_A}(T));\ \mathrm{MAX\_M}(\mathrm{Rnd\_A}(T))],$$

где $\mathrm{Rnd\_A}$ – функция, приводящая в тетракоде-аргументе тетрит A к двоичным 0 или 1, которые выбираются случайным образом.

5.2 Преобразователь вещественных чисел в постбинарные форматы чисел с плавающей запятой

Проектирование и разработка программы «Преобразование в постбинарные форматы чисел с плавающей запятой» (PostBinary format converter, PFC), работающей с постбинарными форматами чисел с плавающей запятой, берет свое начало в целом цикле исследований, подтверждающих эффективность перехода к постбинарным форматам [68, с. 121, 80, с. 240, 91, 115]. В ходе работы над постбинарной арифметикой встали задачи, при которых необходимо представлять вещественные числа в постбинарных форматах в виде набора двоичных полей знака, порядка и мантиссы, и, наоборот, из полей постбинарных форматов получать точное (без округления) представление вещественных чисел в виде десятичной дроби [138].

Созданная программа способна корректно обрабатывать вещественные числа, дроби (определяемые вещественными числителем и знаменателем) и интервалы (определяемые его границами – вещественными числами). Входные числа можно задавать практически любой длины – установленный по умолчанию предел в 1 024 десятичных цифры можно расширять, ограничиваясь лишь объемом свободной оперативной памяти компьютера.

На рисунке 5.4 представлено рабочее пространство программы PFC и демонстрация работы: рассчитана точность представления чисел в формате pb256/128f, которые были получены из входной дроби

+5,877468951515e+3339 / 5,877468951515e–1339

с получением точного отображения десятичных чисел, извлеченных из полей формата, а также расчет абсолютной погрешности между входными и извлеченными значениями. Структура и основные возможности представленного на рисунке 5.4 программного интерфейса:

1) Ввод исходных данных – ввод десятичных чисел, подлежащих преобразованию (поддерживается ввод чисел в нормализованном виде).

Возможен ввод исходных данных в текстовое поле как непосредственно из клавиатуры так и из файла.

2) Блок управления, состоящий из кнопок «Старт» – начало преобразования, «Обнулить» – обнуление всех полей, возврат в исходное состояние, «Пересчитать» – получение точного значения из отредактированных битовых полей форматов минуя поле исходных данных.

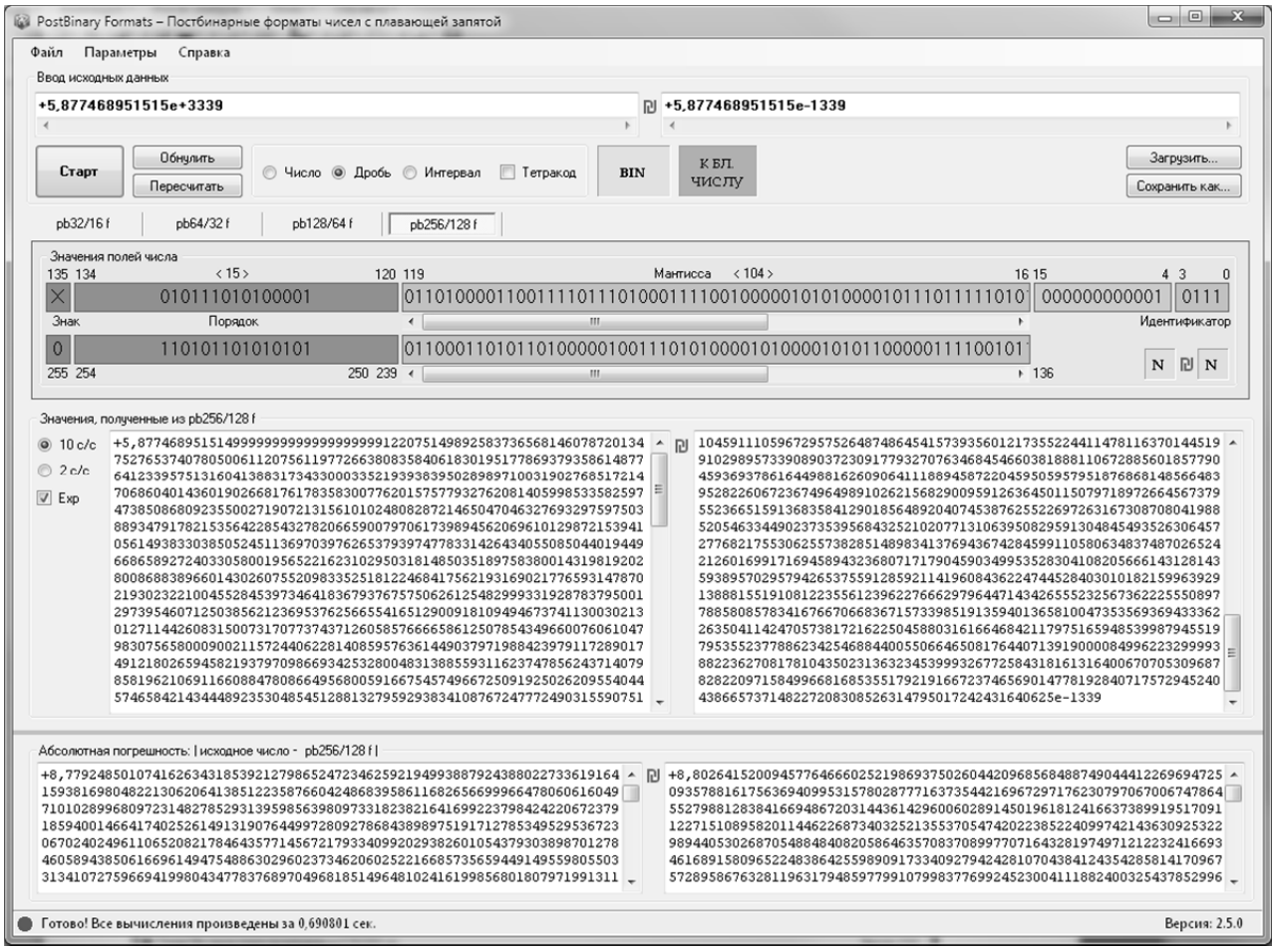


Рисунок 5.4 – Преобразователь PFC (обновленная версия) в режиме точного отображения десятичных чисел, извлеченных из полей формата pb256/128f

3) Вкладки постбинарных 32-, 64-, 128- и 256-разрядных форматов, название которых изменяется в зависимости от состояния компонентов 9 и 11 (Таблица 5.2). Активная вкладка определяет текущий формат, в котором представлено исходное число.

4) Информационное поле формата, содержащее битовые поля знака, порядка и мантиссы с указанием номеров разрядов и количества бит в каждом поле.

5) Режим отображения числа: 10c/c, 2c/c – десятичная и двоичная система счисления; Exp – нормализованный вид числа. Работа данного компонента отображена на рисунке. 5.5.

6) Значение погрешности представления исходного числа в текущем формате (выводится всегда в нормализованном виде). Фактически это разность по модулю значений исходного числа и точного его представления, полученного из полей текущего формата.

7) Переключатель формата входных данных (Таблица 5.2).

8) Компонент для постбинарного кодирования. Его включение означает, что информационные поля формата содержат тетракод (в названии формата появляется суффикс «p») и каждая пара бит поля представляет один четверичный разряд – тетрит.

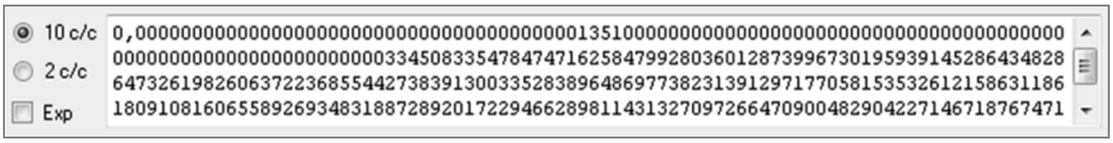
9) Текущий способ округления, установленный на момент преобразования. Поддерживаются как стандартные способы округления чисел стандарта IEEE754 (к нулю, к ближайшему числу, к положительной и отрицательной бесконечности), так и рассмотренное в 4 главе постбинарное округление.

10) Поле идентификатора текущего формата – служебная часть постбинарных форматов, состоящая из полей модификатора и кода формата [68, с. 203].

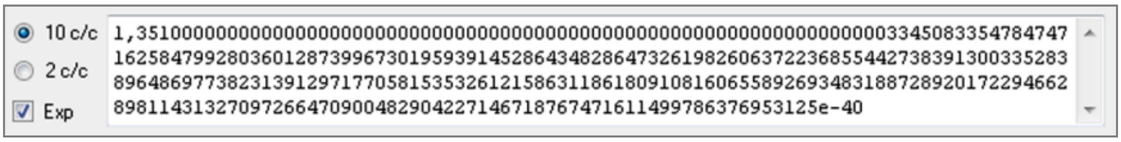
11) Поле точного значения. Отображает результат извлечения из полей текущего формата без потери точности (Рисунок 5.5). В данном поле отображается результат преобразования двоичного числа в десятичную дробь без ограничений вычислительных возможностей процессора и округления чисел операционной системой.

Таблица 5.2 – Постбинарные форматы в зависимости от выбора типа входных данных и способа кодирования

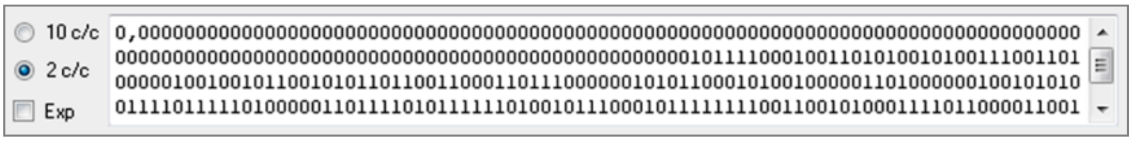
Число	Дробь	Интервал	Тетракод	Разрядность постбинарного формата			
				32	64	128	256
◉	○	○	☐	pb32	pb64	pb128	pb256
			☑	pb32/16p	pb64/32p	pb128/64p	pb256/128p
○	◉	○	☐	–	pb64/32f	pb128/64f	pb256/128f
			☑	–	–	pb128/32fp	pb256/64fp
○	○	◉	☐	–	pb64/32i	pb128/64i	pb256/128i
			☑	–	–	pb128/32ip	pb256/64ip



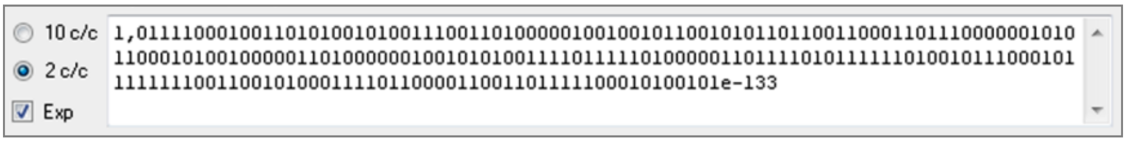
а)



б)



в)



г)

Рисунок 5.5 – Вывод точного значения числа из полей формата pbinary128 (формы вывода числа: *а* – десятичное число; *б* – десятичное нормализованное; *в* – двоичное число; *г* – двоичное нормализованное)

На рисунке 5.6 представлена структурная модель преобразователя постбинарных форматов PFC.

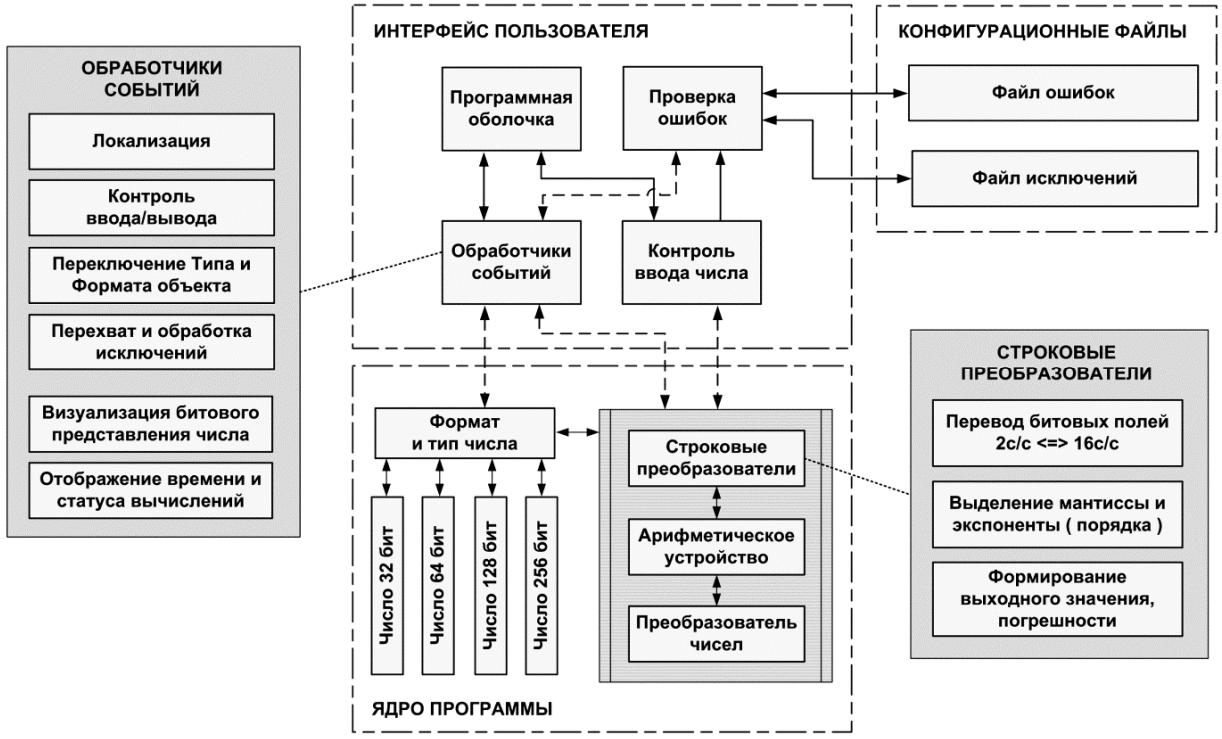


Рисунок 5.6 – Структурная модель PFC (сплошная стрелка – передача входных и выходных параметров, пунктирная стрелка – только выходных параметров)

Ниже приведено краткое описание функциональности блоков модели. Интерфейс пользователя (User Interface, UI) включает в себя набор функций, составляющих каркас программы, а также связывает все остальные блоки с основной формой программы (Рисунок 5.4). Ядро программы – комплекс классов и функций, являющихся основой всех обработок, проверок и преобразований входного числа. Содержит математический аппарат, способный проводить арифметические операции над числами, представленными в виде строк (для реализации длинной арифметики [150, 151, 152]). При этом для реализации такой арифметики используется числовой тип BigInteger пространства имен System.Numerics [153, 154], являющийся сложным целочисленным типом, поддерживающим произвольное число больших целых чисел (не имеет нижней или верхней границы и может содержать любое целочисленное значение).

На основании проведенных исследований при замерах времени обработки чисел различных порядков в постбинарные форматы разной точности типа хранимых чисел, можно сделать ряд заключений:

1) Временные затраты существенно увеличиваются при обработке чисел с возрастающей отрицательной степенью десятичной экспоненты. Иными словами, чем исходное число «ближе к нулю», тем больше времени требуется для его обработки. Это связано с операциями над числами, представляющими большие отрицательные степени двойки. При этом большую часть времени занимает вычисление точного значения этих чисел.

Например, при исходном числе 1,5e–200 для получения точного значения, извлеченного из полей форматов, производится преобразование следующих двоичных дробей:

- для pbinary64:

```
1,0010010111101110110110001111111110110011100111100e-664;
```

- для pbinary128:

```
1,00100101111011101101100011111111101100111001110000010001011110
1100011110010010011111101001100110000011010e-664;
```

- для pbinary256:

```
1,00100101111011101101100011111111101100111001110000010001011110
1100011110010010011111101001100110000011001100010011100111111101
0110111001101011111111101010010101101011111110010011011011010001
0101011001000100011011110010011e-664
```

в соответствующие десятичные дроби:

- для pbinary64:

```
1,49999999999999968306684161266835577666944638341229562731698860
4535518717844272789199436274191888833431534392330820717096774434
6305706583068382429364958665499574673400370351047036375195939993
1584075683170771351133154899405913582310844055191206758665801192
8458125402280264071552132222601209372805671713096801257681381538
0685440020326456171347353241477997512793597703292517866507460278
1932834164429876742735095516552396076727026791921947891777325038
4519098398567994601267815146794504244098789058625698089599609375
e-200;
```

- для pbinary128:

```
1,5000000000000000000000000000029736925385879603512981657336393886180844159622543222587667348491274409135991825637900579782014114689514001626100114350697931034298947399662706411139733715495495717201445294803323000740894900404976411385058911077819153397307301220041872674712966875057862554268072845031235122527745649807629541176195615183891700838589512934935300732062585961874920218562328773990028956077894356832191218416630769452484341896155337332103079976056754911538113122610436374545394080319648658670535054051412306221091087776731143321740091778337955474853515625e-200;
```

- для pbinary256:

```
1,4999999999999999999999999999999999999999999999999999999999999999999999982196704765218111393475202830367545985687123199423459681447466848066948863511282043888954023549664869965327127303020313596213436760436305857080620690580672100607213236194344171866467422297439560461273736196341507418363558985582382478249298925161067583559513607854125325581150168131634044310336491291374096096618689331619858464991533829036717321144420911536844767704099632875001045926275065877616845818337228719704425692510666990330049456470295601275873300665895365898081504965465734685505291741738399750833713348056839472052724443295165741992065325454077336677868196276566860320213949577805578883271664381027221679687 5e-200.
```

Данный пример демонстрирует вычислительные возможности программы. Причем для получения значения погрешности представления чисел организована операция вычитания по модулю между извлеченным из полей формата и исходным числами.

2) На основании предыдущего пункта: денормализованные числа требуют для обработки и преобразования на порядок больше времени, чем нормализованные числа.

3) Моменты времени t, необходимые для полного цикла преобразований одного входного числа в форматах различной точности имеют экспоненциальную зависимость, пропорциональную разрядности формата. То есть, чем точнее формат (больше разрядность), тем больше времени необходимо для работы с ним. Очевидно, что

$$t(\text{pbinary32}) < t(\text{pbinary64}) < t(\text{pbinary128}) < t(\text{pbinary256}).$$

Рассмотрим варианты преобразования числа на примере числового, дробного и интервального форматов с получением погрешности представления. В качестве исходного числа рассмотрим математическую константу e –

иррациональное и трансцендентное число, основание натурального логарифма, – играющее важную роль в дифференциальном и интегральном исчислении, а также во многих других разделах математики. Иррациональность данного числа указывает, что его невозможно точно представить в виде десятичной и рациональной дробей. Поэтому воспользуемся десятичным и рациональным приближениями с погрешностью, например, не более 10^{-9}. Получаем значения входных данных в виде

- числа $e \approx 2{,}718281828$;
- рациональной дроби $e \approx \frac{271801}{99990}$;
- интервала $e \in [2{,}718281828;\ 2{,}718281829]$.

На рисунке 5.7 приведены битовые значения полей соответствующих типов на примере 128-разрядных форматов (для данного примера выбраны типы форматов, использующих бинарное кодирование), а также значения, извлеченные из данных полей. В таблице 5.3 сведены погрешности представления исходных данных для каждого формата.

На основании полученных значений из полей форматов и величины погрешности можно сделать следующие заключения:

1) При представлении иррационального числа в числовом формате, погрешности исходного числа и его представления в виде числа с плавающей запятой накладываются, что может привести к еще большей потере точности (в нашем случае в формате pb32 погрешность исходного числа снизилась с 10^{-9} до 10^{-7}).

2) Целые числа числителя и знаменателя рациональной дроби представляются без погрешности (с учетом, что данные числа входят в диапазон представления формата). При этом сохраняется только исходная погрешность числа – погрешность рационального приближения.

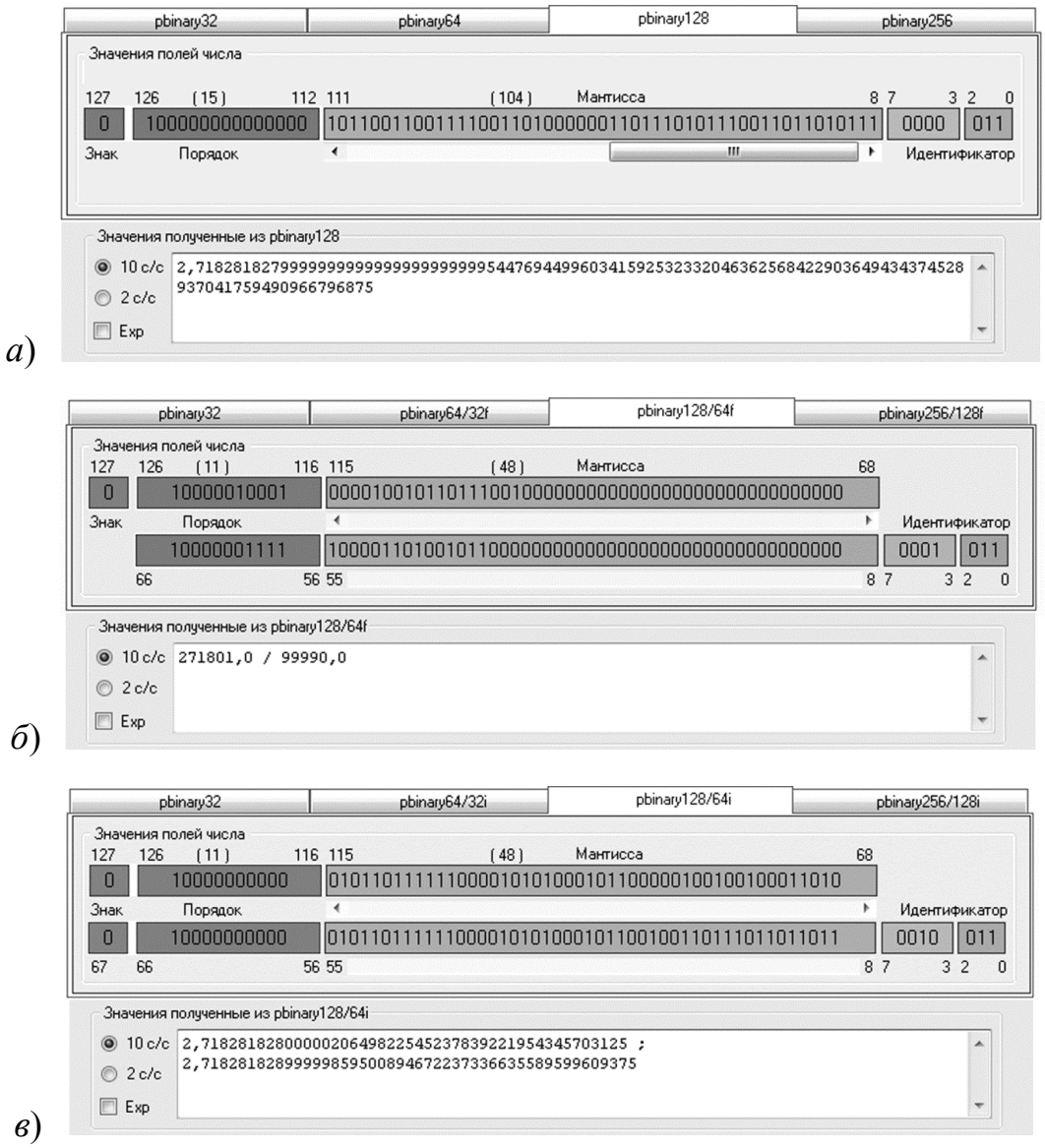


Рисунок 5.7 – Представление константы e в 128-разрядных постбинарных форматах в виде десятичной (*а*) и рациональной (*б*) дробей, а также в виде вещественного интервала (*в*)

3) Интервальное представление исходного числа имеет преимущество, поскольку заданный интервал содержит исходное число независимо от класса точности последнего. Однако при задании границ интервала следует учитывать относительную точность десятичных цифр, которую обеспечивают форматы с плавающей запятой. Эта точность определяется по формуле $\left(\lg 2^{m+1}\right) \in \mathbb{Z}$, где m – количество разрядов мантиссы. В нашем случае для 64-разрядного интервального

формата количество разрядов мантисс для каждой границы равно 21, следовательно, относительная точность равна $\lg 2^{22} = 7$ десятичных цифр.

Таблица 5.3 – Погрешность представления константы *e* в числовых дробных и интервальных форматах

Разрядность	Число	Дробь	Интервал
32	8,208935547e–7	–*	–
64	2,064982255e–14	0,0 / 0,0	8,208935547e–8; 8,308935547e–8
128	4,552305500e–31		2,064982255e–15; 1,4049910533e–15
256	2,375392996e–66		4,552305500e–31; 8,608605556e–33

* – работа с дробями и интервалами не предусмотрена (см. приложение И)

5.3. Анализ и исследование интервальных операций при переходе к постбинарным форматам с плавающей запятой

В предыдущих главах диссертационного исследования детально рассмотрены анализ, принципы и методы качественно нового подхода в кодировании действительных чисел в памяти компьютерных систем и выполнении последующих арифметических операций над ними. Это выражено получением модифицированных форматов чисел с плавающей запятой, которые названы постбинарными (подробная спецификация модифицированных форматов приведена в [80, с. 282–307], а также в приложении И, а порядок выполнения арифметических операций − постбинарной арифметикой. Такая арифметика, определяющая собственные методы и средства вычислительных процессов, воспроизводимая на соответствующих компьютерных компонентах является составной частью постбинарного компьютинга. Концепция последнего заключается в том, что в качестве способа представления данных выступает

тетракод, представленный комбинацией тетритов. При этом тетрит, как разряд тетракода, является единицей измерения количества информации и может принимать значения тетрануля (0), тетраединицы (1), неопределенности (A) и множественности (M) [68, с. 15], [83, 85], [87, с. 74]. Далее показаны преимущества использования постбинарных форматов над бинарными, так как первые приводят к достоверным вычислениям, в то время, когда использование стандартных (бинарных) форматов с плавающей запятой допускает необратимые погрешности, как при кодировании исходных чисел, так и при выполнении арифметических операций, что приводит к неверным (неточным) результатам.

Рассмотрим пошаговое вычисление выражения $B \cdot C - D$ интервальных чисел B, C и D, представленных в постбинарном интервальном формате pb128/32ip [80, с. 306] (приложение И) при следующих интервальных значениях:

$$B = [9.258\text{e}{-6},\ 9.25805\text{e}{-6}]; \quad C = [-0.12484,\ -0.07031249];$$

$$D = [4.89\text{e}{-8},\ 5.681067\text{e}{-8}].$$

С учетом особенности стандартного формата одинарной точности (точность до 23 двоичных или 7 десятичных значащих цифр):

$$B \cdot C - D = [-1.21259\text{e}{-6}, -6.99853\text{e}{-7}].$$

Расчет в СКА Mathematica 7.0.1 [130, 155] с заданной точностью до 16 значащих цифр представлен на рисунке 5.8.

```
In[126]:= B1 = Interval[{9.258 * 10^-6, 9.25805 * 10^-6}];
          C1 = Interval[{-0.12484, -0.07031249}];
          D1 = Interval[{4.89 * 10^-8, 5.681067 * 10^-8}];
          SetPrecision[B1 * C1, 16]
          SetPrecision[% - D1, 16]

Out[129]= Interval[{-1.155774962000001 × 10^-6, -6.509530324199995 × 10^-7}]

Out[130]= Interval[{-1.212585632000001 × 10^-6, -6.998530324199992 × 10^-7}]
```

Рисунок 5.8 – Результаты вычислений в СКА Mathematica

Стандартный калькулятор Windows 7 (версия калькулятора 6.1, сборка 7601: Service Pack 1) с заданной аналогичной точностью возвращает следующие значения интервалов:

$$B \cdot C = [-0{,}000001155774962, -0{,}00000065095303242];$$

$$B \cdot C - D = [-0{,}000001212585632, -0{,}00000069985303242].$$

На данном этапе вполне очевидно, что полученные числа, а, следовательно, и промежуточные значения, подверглись округлению, причем в каждой программе выводятся по-разному уточненные результаты. Таким образом, можно утверждать, что в процессе вычислений с плавающей запятой, результаты в большинстве случаев получены с определенной погрешностью, которую невозможно контролировать на всех этапах компьютерных вычислений. Так, в среде Mathematica и на калькуляторе Windows, получены не эквивалентные значения. Арифметика чисел в формате pb128/32ip позволяет это исправить. Следует отметить, что представленный в [80, с. 306] формат pb128/32ip предназначен для хранения двух вещественных границ интервала в виде тетракода.

Интервал $X = [x_1, x_2]$, где x_1 и x_2 – левая и правая границы интервала соответственно: $X = \{x_i \mid x_1 \le x_i \le x_2, \ x_i \in \mathbb{R}\}$ [68, с. 108]. На рисунке 5.9 представлен интервал $X = [0, 0]$ при x_1 и x_2 равными нулю в формате pb128/32ip в тетритах (следует учитывать, что содержимое полей MF и CF всегда битовое).

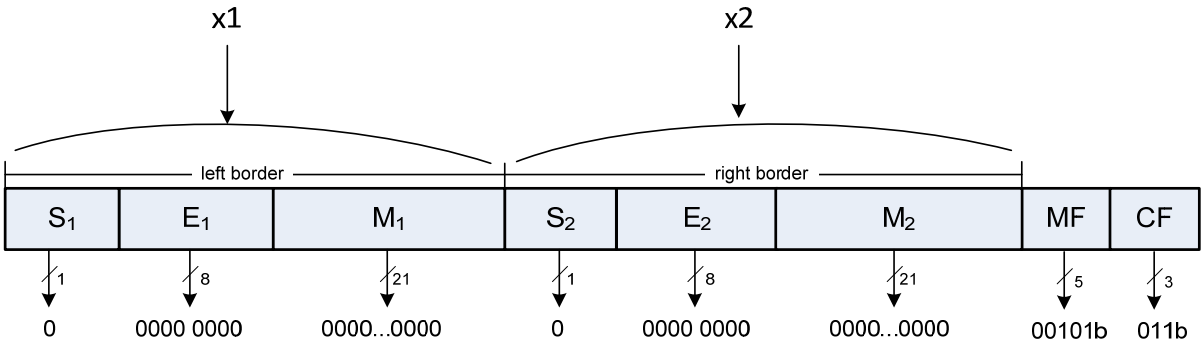


Рисунок 5.9 – Интервал $X = [0, 0]$ в формате pb128/32ip

Поскольку один тетрит в памяти бинарного компьютера представлен парой бит, а именно: 0t → 01b, 1t → 10b, At → 00b, Mt → 11b [97, с. 237], то нулевой интервал $X = [0, 0]$ представляется в битах для формата pb128/32ip так, как это показано на рисунке 5.10 (т. е. это уже 128-разрядная битовая последовательность).

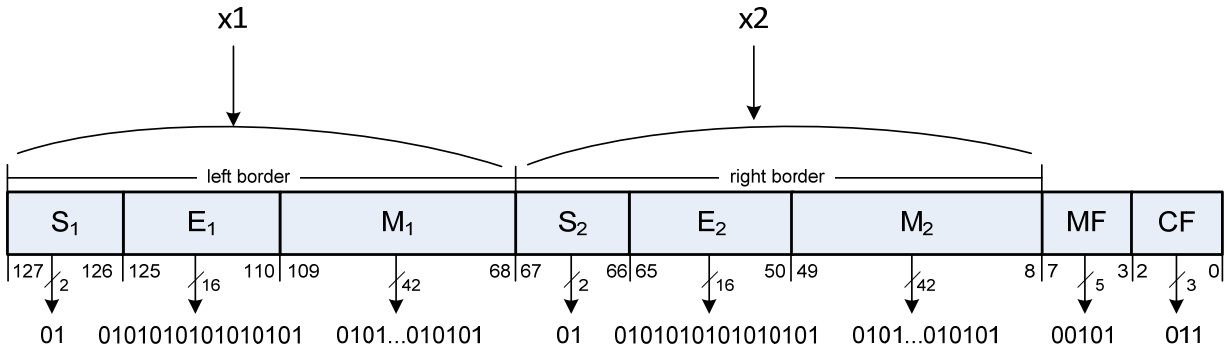


Рисунок 5.10 – Битовая последовательность постбинарного формата pb128/32ip, задающая границы интервала $X = [0, 0]$

Перевод исходных значений в указанный формат. В постбинарных форматах (на что указывает наличие буквы «p» в обозначении формата) применяется по умолчанию постбинарное округление, а в двоичных интервальных форматах (в названии присутствует только буква «i», без буквы «p») применяется интервальное округление (левая граница округляется к отрицательной бесконечности, правая – к положительной бесконечности). В остальных форматах округление по умолчанию – к ближайшему числу, как принято стандартом IEEE 754-2008 [54].

При этом порядок перевода чисел в постбинарные форматы можно свести к следующим шагам:

1) Перевод исходного числа согласно стандарту IEEE 754 в формат той точности, которая указана после слэша «/» в обозначении постбинарного формата (для pb128/32ip это формат одинарной точности pb32 [68, с. 206]) с запоминанием двух выпавших старших разрядов за пределы поля мантиссы. Значения полей формата задаются в тетритах $T \in \{0, 1\}$, которые в данном случае эквивалентны

битовому множеству $B \in \{0, 1\}$, поскольку на этом шаге не встречаются значения A и M.

2) Анализируя выпавшие разряды, применяются правила постбинарного округления [66, с. 211] над мантиссой формата. Здесь множество T включает четыре состояния $T \in \{0, 1, A, M\}$.

3) Полученный тетракод в битовом представлении записывается в поля формата таким образом, чтобы ширина поля соответствовала числу, указанному в названии формата до слэша «/» (для pb128/32ip это поле 128 бит, куда укладываются два комплекта формата pb32 – левая и правая границы интервала). При этом, поскольку тетрит кодируется парой бит, n-разрядный тетракод можно записать $2n$-разрядным битовым кодом.

4) В случае составного формата (в обозначении присутствуют буквы «i» или «f») шаги 1–3 повторяются для второго числа интервала (правая граница) или дроби (знаменатель). При этом значение левой границы интервала или числитель дроби записываются в старшие разряды формата, а значение правой границы интервала или знаменатель дроби записываются в младшие разряды формата;

5) В поля модификатора и кода формата записываются соответствующие битовые коды [68, с. 205].

Применив указанные выше шаги для исходных значений контрольного примера, получаем поля форматов, представленных на рисунках 5.11–5.13.

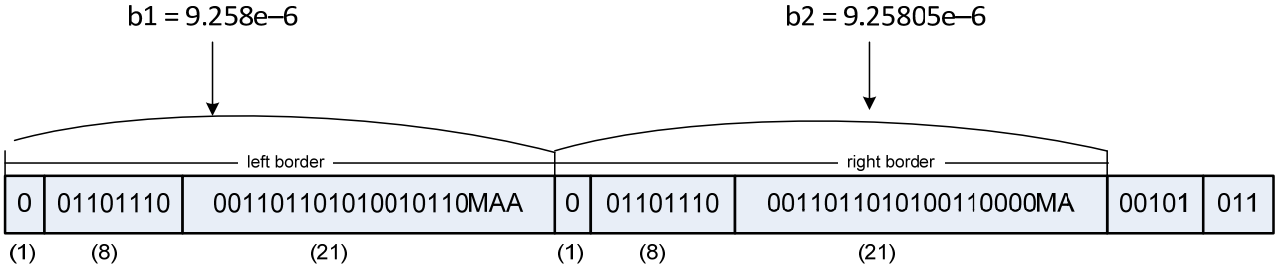


Рисунок 5.11 – Интервал B = [9.258e–6, 9.25805e–6] в формате pb128/32ip

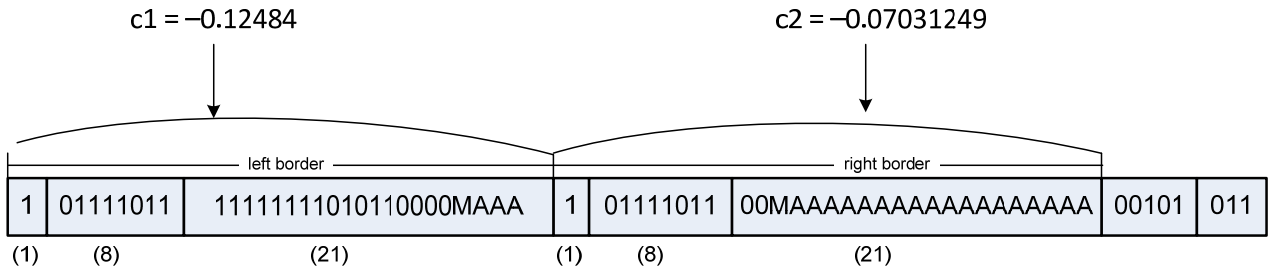


Рисунок 5.12 – Интервал C = [–0.12484, –0.07031249] в формате pb128/32ip

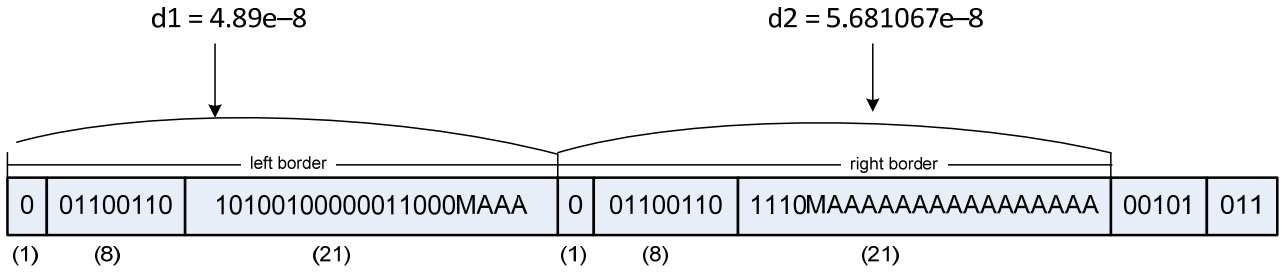


Рисунок 5.13 – Интервал D = [4.89e–8, 5.681067e–8] в формате pb128/32ip

Выполним операцию умножения интервалов. По правилам интервального умножения:

$$BC = [b_1, b_2] \cdot [c_1, c_2] = [\min(b_1c_1, b_1c_2, b_2c_1, b_2c_2), \max(b_1c_1, b_1c_2, b_2c_1, b_2c_2)],$$

т. е. необходимо сделать четыре умножения и выбрать максимальное (левая граница) и минимальное (правая граница) значения для формирования формата pb128/32ip со значением BC.

Таким образом,

$$BC = [\min(9.258\text{e–}6 \times (–0.12484),\ 9.258\text{e–}6 \times (–0.07031249),$$
$$9.25805\text{e–}6 \times (–0.12484),\ ),\ 9.25805\text{e–}6 \times (–0.07031249)),$$
$$\max(9.258\text{e–}6 \times (–0.12484),\ 9.258\text{e–}6 \times (–0.07031249),$$
$$9.25805\text{e–}6 \times (–0.12484),\ ),\ 9.25805\text{e–}6 \times (–0.07031249))\ ].$$

Порядок умножения чисел в постбинарных форматах с плавающей запятой сводится к следующим пошаговым действиям:

Шаг 1. Сложение смещенных порядков множителей в полном соответствии со стандартом IEEE 754-2008, как числа в формате с отрицательным нулем [66, с. 232].

Шаг 2. Проверка суммы (порядка результата) на переполнение также согласно правилам стандарта IEEE 754-2008. В случае обнаружения переполнения:

а) фиксируется сигнал переполнения. В формате результата формируется значение бесконечности (в поле мантиссы все разряды заполняется тетранулями, в поле порядка – тетраединицами). После осуществляется переход к шагу 6, в котором формируется соответствующий знак бесконечности;

б) выполнение алгоритма перехода к форматам следующего класса точности [66, с. 237–238]. В данном случае это переход из pb128/32ip к pb256/64ip. Далее переход к шагу 1 для повторения операции умножения чисел уже в новых форматах.

Шаг 3. Выполнение умножения мантисс (с учетом скрытых единиц) и последующая проверка нарушения нормализации мантиссы результата.

Шаг 4. Если нормализация нарушена (а она может быть нарушена только влево), то:

а) производится восстановление мантиссы, т. е. выполнение сдвига мантиссы результата вправо с инкрементом (увеличением на 1) порядка результата;

б) выполняется проверка порядка после инкрементирования на переполнение (см. Шаг 2).

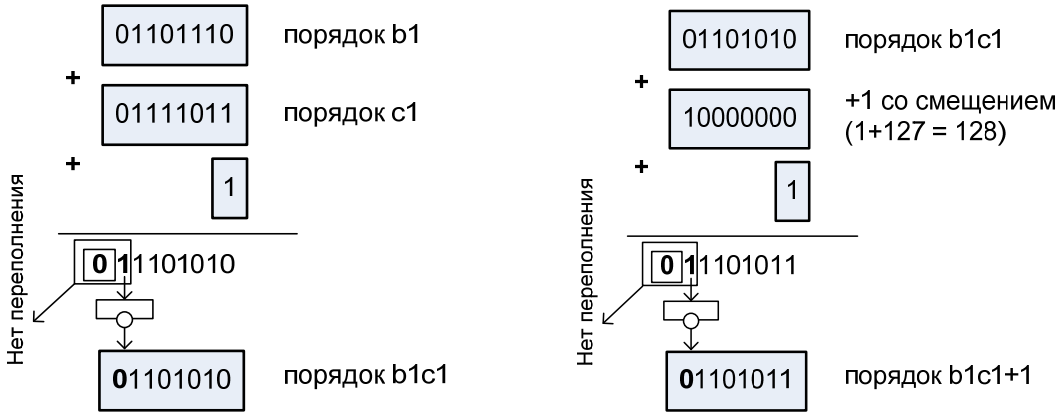
Шаг 5. Выполнение постбинарного округления мантиссы результата. Следует отметить, что при округлении в данном случае рассматривается больше ситуаций (все ситуации, возникающие при выполнении данного округления, а также коррекция мантиссы, показаны при выполнении примера на рисунке 5.15), чем в случае процедуры округления при исходном формировании формата (см. рисунки 5.11–5.13).

Шаг 6. Формирование знака результата как суммы по модулю 2 знаков множителей.

На основании приведенного порядка умножения чисел в постбинарных форматах с плавающей запятой, выполним умножение $b_1 c_1 = (9.258\ \text{e} - 6) \times (-0.12484)$:

Шаг 1. Сложение порядков чисел b_1 и c_1 (Рисунок 5.14 *а*).

Шаг 2. Проверка переполнения суммы порядков. Переполнение не обнаружено, поскольку выходной перенос при сложении не равен старшему разряду порядка-результата.



а) *б*)

Рисунок 5.14 – Сложение порядков форматов чисел b_1 и c_1 (*а*) и инкремент полученной суммы (*б*)

Шаг 3. Умножение мантисс (в приложении К показан пример ручного умножения тетритов $b_1 c_1$ в «столбик»). При этом следует учитывать, что для тетракодов, согласно рассмотренным ранее правилам умножения тетритов, справедливы следующие равенства: $1 \times x = x$, $0 \times x = 0$, $\text{A} \times \text{A} = \text{A}$, $\text{M} \times \text{M} = \text{M}$, $\text{A} \times \text{M} = 0$, где $x \in \{0, 1, \text{A}, \text{M}\}$ [80, с. 158–161].

Шаг 4. Проверка нарушения нормализации мантиссы результата. В примере умножения мантисс (приложение К) продемонстрировано нарушение нормализации мантиссы результата. Следовательно, мантисса сдвигается на один разряд вправо, а порядок увеличивается на единицу (Рисунок 5.14 *б*).

Шаг 5. Проверка переполнения порядка результата после увеличения на 1 при нормализации мантиссы. В поле порядка переполнение отсутствует (см. рисунок 5.14 *б*).

Шаг 6. Выполнение округления мантиссы (Рисунок 5.15).

Шаг 7. Формирования знака результата (Рисунок 5.15).

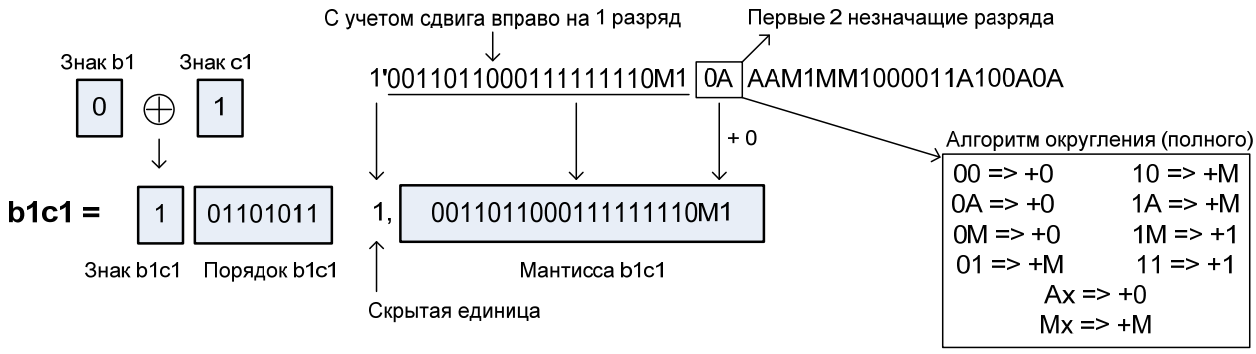


Рисунок 5.15 – Выполнение округления мантиссы и формирование знака результата произведения b_1c_1

Результаты остальных умножений (b_1c_2, b_2c_1, b_2c_2) приведены на рисунке 5.16. Чтобы получить результирующий интервал BC, необходимо определить минимальное и максимальное значение полученных произведений, т. е. получить $\min(b_1c_1, b_1c_2, b_2c_1, b_2c_2)$ и $\max(b_1c_1, b_1c_2, b_2c_1, b_2c_2)$ для последующего заполнения полученными значениями старшую и младшую половину интервального формата. Для этого необходимо выполнить сравнение полученных значений.

Порядок и принцип сравнивания чисел с плавающей запятой (числа одного знака) заключается в следующем:

1) Сравнение проводится над модулями форматов (без учета знака).
2) Сравниваются порядки: большее значение имеет бóльший порядок.
3) Если порядки равны – сравниваются мантиссы: большее число имеет бóльшую мантиссу.

4) Учет знака. Если он равен 1 (для отрицательных чисел): найденный максимальный модуль является модулем меньшего числа, а найденный минимальный модуль – модулем большего числа.

Для текущего примера, с учетом значений порядка целесообразно сравнивать:

– модули мантисс b_1c_1 и b_2c_1 (имеют одинаковый порядок 01101011);

– модули мантисс b_1c_2 и b_2c_2 (имеют одинаковый порядок 01101010).

При этом большее значение модуля из пары b_1c_1 и b_2c_1 будет максимальным (правой границей результирующего интервала), так как числа имеют бóльший порядок, поэтому меньшее значение модуля из пары b_1c_2 и b_2c_2 будет минимальным (левая граница результирующего интервала).

Поиск модуля максимального и минимального значения схематически показан на рисунке 5.17. Следовательно, с учетом знака:

$$\min(b_1c_1, b_1c_2, b_2c_1, b_2c_2) = b_2c_1;$$

$$\max(b_1c_1, b_1c_2, b_2c_1, b_2c_2) = b_1c_2.$$

Таким образом, интервал BC в формате pb128/32ip представлен на рисунке 5.18. При этом в памяти компьютера формат со значением BC представляет собой следующую 128-битную последовательность (разряды знака выделены одинарным подчеркиванием, поля порядков – полужирным начертанием, модификатор и код формата – двойным подчеркиванием):

<u>1</u>0 **0110100110011010** 01011010011010010110010101010101011100001 0

<u>1</u>0 **0110100110011001** 01100110101001100110101001100101100111111 <u>00101011</u>

Далее выполняется операции вычитания. По правилам интервального вычитания:

$$BC - D = [bc_1, bc_2] - [d_1, d_2] = [bc_1 - d_2, bc_2 - d_1],$$

т. е. необходимо сделать два вычитания для левой и правой границы результата в формате pb128/32ip.

При этом порядок вычитания (сложения) чисел в постбинарных форматах с плавающей запятой сводится к следующим пошаговым действиям:

Шаг 1. Выполняется выравнивание порядков: меньший порядок приближается по значению к большему путем инкрементирования (следовательно, мантисса этого числа сдвигается вправо на один разряд при каждом увеличении значения порядка на 1) пока не станет равным большему порядку.

Шаг 2. Выполняется постбинарное округление мантиссы с учетом «выпавшей части» из поля мантиссы для числа, чей порядок подвергся выравниванию.

Шаг 3. Выполняется соответствующая арифметика над мантиссами. В этом случае выполняется *алгоритм выбора операции* в зависимости от знаков операндов, вида операции (сложение или вычитание) и результата сравнения модулей мантисс операндов, а также проверяется нарушение нормализации мантиссы суммы (разности).

Шаг 4. Если зафиксировано нарушение нормализации мантиссы влево, то происходит ее восстановление – сдвиг мантиссы результата вправо (пока скрытая единица не займет свое место) и инкремент порядка результата. При этом все действия в случае переполнения порядка и округления мантиссы аналогичны подобным действиям при рассмотренном ранее умножении чисел.

Шаг 5. Если нормализация нарушена вправо, то выполняется восстановление путем сдвига мантиссы результата влево, пока старшая единица мантиссы не займет место скрытой единицы (в освободившиеся разряды заносится значение «А», если младший разряд мантиссы был «А», или «0» – в остальных случаях) и декремент порядка результата. Действия в случае переполнения порядка и округления мантиссы также аналогичны подобным действиям при рассмотренном ранее умножении чисел.

Шаг 6. Формирование знака результата согласно алгоритму выбора операции (см. шаг 3).

Рисунок 5.16 – Форматы результатов умножения чисел b_1, c_1, b_2 и c_2

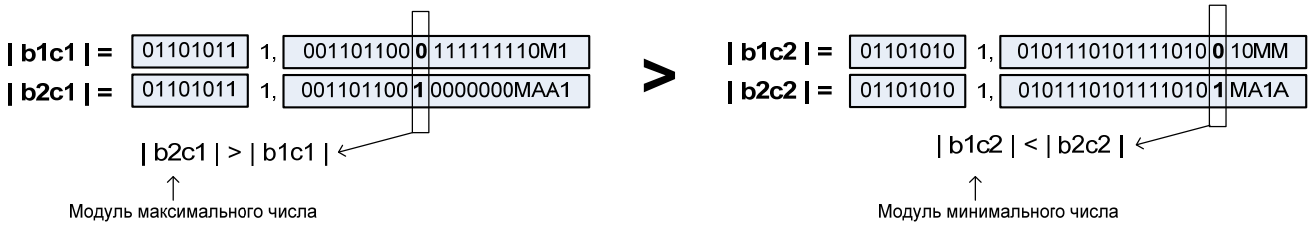


Рисунок 5.17 – Поиск модуля максимального и минимального чисел

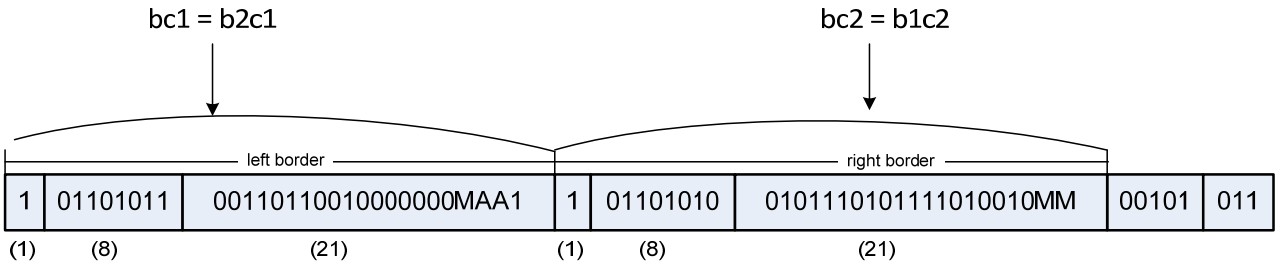


Рисунок 5.18 – Интервал *BC* в формате pb128/32ip

Рассмотрим вычитание $bc_1 - d_2$, значения полей формата которых приведены на рисунке 5.19. Из рисунка видно, что порядок d_2 меньше порядка bc_1 (с помощью аппаратных средств это можно определить по стандартной схеме сравнения [66, с. 239]). Определим количество сдвигов вправо для мантиссы d_2 путем вычитания порядков (Рисунок 5.20). При вычитании учитывается, что смещенные порядки в стандарте IEEE 754-2008 представлены в формате с отрицательным нулем. Полученная разность порядков равна 5. Это значит, что 21-разрядную мантиссу числа d_2 нужно сдвинуть на 5 разрядов вправо.

Иногда полученное значение разницы порядков может быть больше разрядности мантиссы, что приводит к *полной потере значимости* мантиссы после выравнивания порядков. Поэтому критерий

(количество_сдвигов_при_выравнивании_порядков) > (разрядность_мантиссы)

можно считать *критерием потери точности* при традиционных вычислениях и основным *критерием для динамического наращивания разрядности* модифицированных постбинарных форматов (т. е. в данном случае переход от формата pb128/32ip к формату pb256/64ip). Однако, в случае потери значимости мантиссы операнда, с учетом принципов тетракодирования и возможности использования постбинарного округления в модифицированных форматах чисел с плавающей запятой, в *худшем случае* (т. е. без возможности наращивания разрядности) можно все равно сохранить значимость результата.

Далее получаем значение d_2 после выравнивания порядков (Рисунок 5.21). Имеем следующие показатели алгоритма выбора операции: исходная операция – вычитание, знак $bc_1 = 1$, знак $d_2 = 0$, мантисса bc_1 больше мантиссы d_2 (т. к. мантисса d_2 уменьшилась после выравнивания порядков). Следовательно, определены *фактическая операция* – операция сложения и *знак результата* – знак числа bc_1. В результате фактической операции, формирование значений полей формата разности $bc_1 - d_2$, схематически показано на рисунке 5.22.

Рисунок 5.19 – Исходные значения полей формата чисел bc_1 и d_2

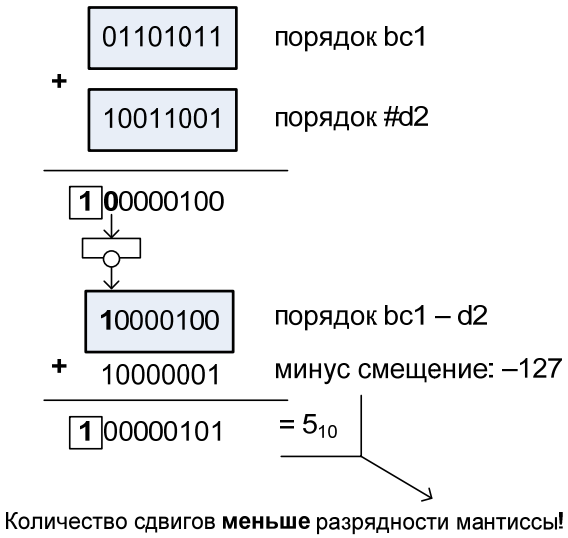


Рисунок 5.20 – Определение разности порядков чисел bc_1 и d_2

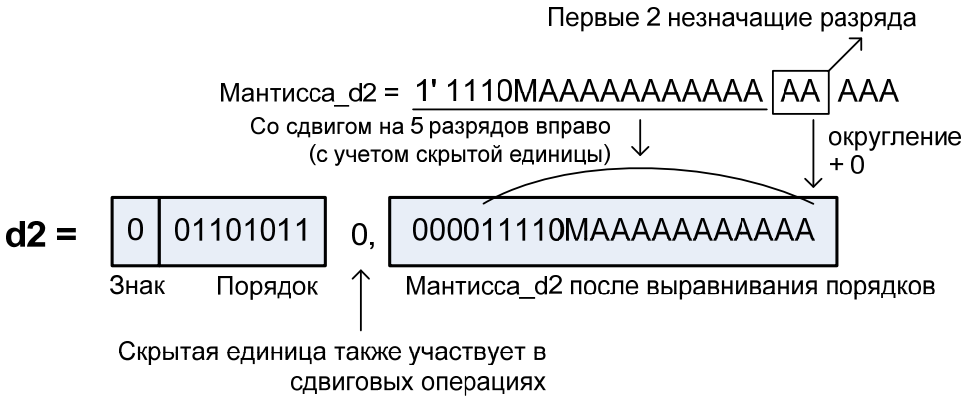


Рисунок 5.21 – Формирование смещенного значения мантиссы числа d_2 после выравнивания порядков

Рисунок 5.22 –Реализация разности $bc_1 – d_2$ после выравнивания порядков

Аналогично производится разность $bc_2 - d_1$, последний этап реализации которой схематически приведен на рисунке 5.23.

Таким образом, полученные разности $bc_1 - d_2$ и $bc_2 - d_1$ позволяют получить итоговое значение тестовых пошаговых вычислений, результатом которых является выражение $BC - D$ в интервальном постбинарном формате pb128/32ip (Рисунок 5.24). Следует также отметить, что в памяти компьютера конечный результат разности интервалов $BC - D$ представляет собой 128-битную последовательность:

10 **0110100110011001** 01100101011001101001010101010101110000011

10 **0110100110011001** 01101010011010101001101010011010010111010100101011

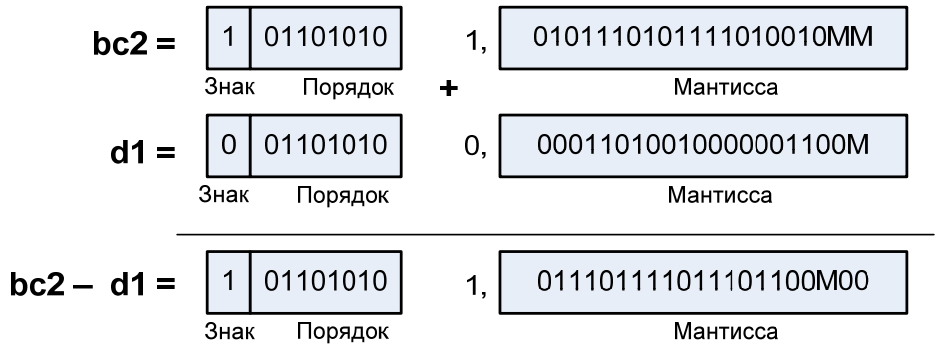


Рисунок 5.23 – Реализация разности $bc_2 - d_1$ с выровненными порядками и фактической операции сложения для мантисс, с формированием знака результата, равного знаку bc_2

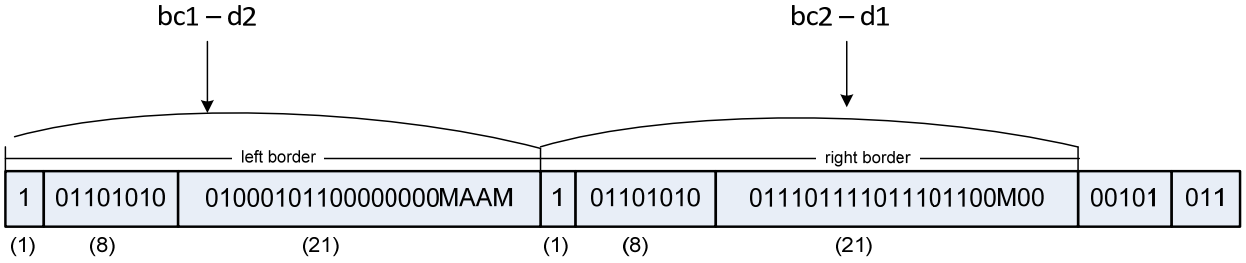


Рисунок 5.24 – Интервал $BC - D$ в формате pb128/32ip

На рисунке 5.25 приведены десятичные результаты, полученные из формата pb128/32ip с результатом $BC - D$. Следует отметить, что, согласно [80, с. 133–

143], постбинарное число возвращает бинарный интервал, границы которого (без учета знака) формируются следующим образом:

– меньшее число (левая или нижняя граница интервала): 0t → 0b, 1t → 1b, Mt → 0b, At → 1b;

– большее число (правая или верхняя граница интервала): 0t → 0b, 1t → 1b, Mt → 1b, At → 0b.

Полученные диапазоны десятичных значений для каждой границы интервала (Рисунок 5.25) при сравнении с исходными расчетными значениями гарантируют достоверность и обеспечивают учет ошибок округления при кодировании чисел и при проведении арифметических операций над ними. Об этом свидетельствует рисунку 5.26, в котором диапазоны границ общего интервала $BC - D$ представлены с учетом знака (для отрицательных чисел: меньшее по модулю число является правой границей интервала). Таким образом, границы полученного интервала сами по себе представляют числовые диапазоны, в которые входят все допущенные погрешности, полученные при контрольном просчете тестового примера.

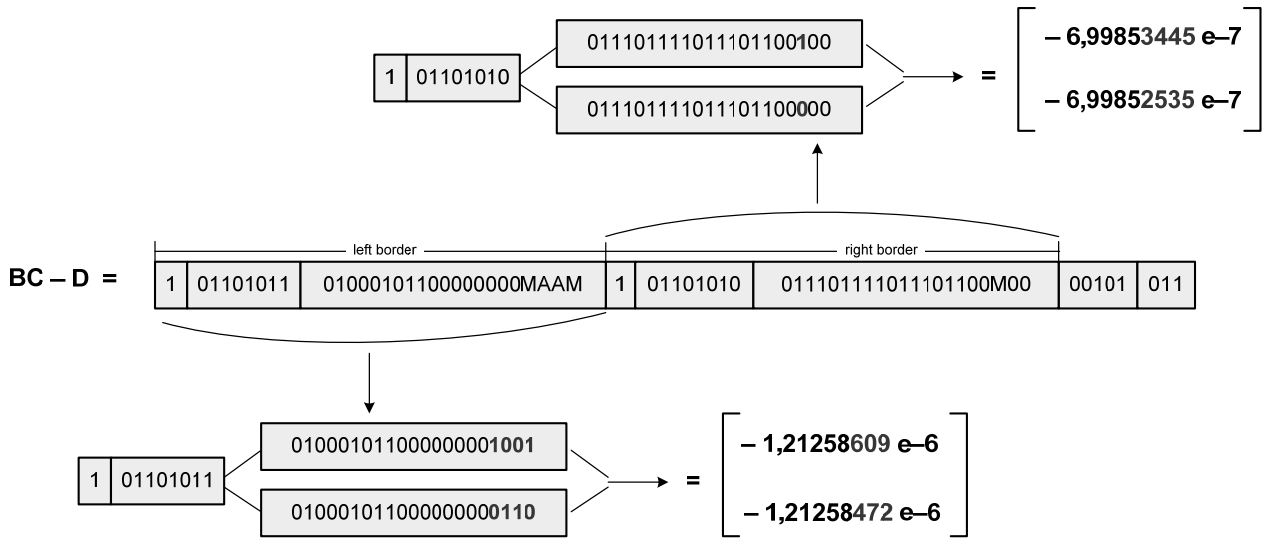


Рисунок 5.25 – Получение десятичных диапазонов для каждой границы интервала-результата $BC - D$, представленного в постбинарном формате pb128/32ip

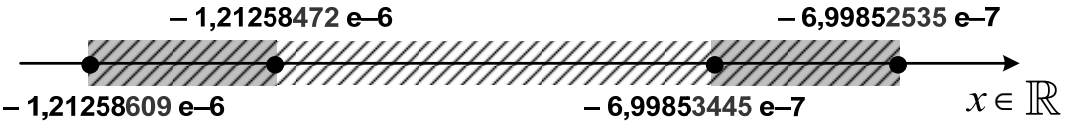


Рисунок 5.26 – Полученный интервал $BC - D$ на числовой шкале с выделенными диапазонами левой и правой границ

В результате компьютерных интервальных вычислений, наряду с интервальным приближением количественных величин, упускаются погрешности, возникающие на этапе кодирования значений-границ интервалов в форматах чисел с плавающей запятой. Поскольку интервальная арифметика оперирует только конечными значениями интервальных чисел, то при каждой операции компьютерных интервальных вычислений возможен неконтролируемый рост погрешности представления границ интервала-результата.

5.4 Выводы по пятой главе

Использование программной реализации кодирования вещественных чисел в постбинарные форматы с плавающей запятой позволило провести исследования, в результате которых выяснилось, что интервальное представление исходного числа имеет преимущество перед иными представлениями (число, дробь). Это связано, прежде всего, с тем, что заданный интервал содержит исходное число независимо от класса точности последнего. Однако при задании границ интервала следует учитывать относительную точность десятичных цифр, которую обеспечивают форматы с плавающей запятой. Постбинарный подход в кодировании интервальных чисел обеспечивает устранение погрешности, которая может возникнуть как при кодировании чисел, так и при выполнении арифметических операций над интервалами.

Предлагаемый переход к постбинарному кодированию, и соответствующим интервальным форматам чисел с плавающей запятой, позволяет, во-первых, хранить границы интервальной переменной в едином поле формата, и, во-вторых,

каждую из границ интервальной переменной представлять в виде интервалов (допусков) для каждой границы. При этом каждая из границ интервальной переменной задана единственным постбинарным кодом и обрабатывается как одно число. Получение результирующих допусков границ интервала можно производить после окончания расчетов при переводе результата в десятичную систему счисления (как видно на рисунке 5.25, такой перевод производится через бинарные представления соответствующих полей формата с плавающей запятой).

В данной главе представлены и выполнены пошаговые алгоритмы сложения, вычитания и умножения тетракодов на примере интервальной арифметики постбинарных форматов с плавающей запятой. Полученные в результате интервальных вычислений границы результирующего интервала при использовании бинарных вычислений обладают определенной погрешностью, однако они принадлежат диапазонам границ полученного постбинарного общего интервала при выполнении соответствующей арифметики с числами в формате pb128/32ip.

ЗАКЛЮЧЕНИЕ

В диссертационной работе дано теоретическое обоснование и приведено решение научно-технической задачи разработки концепции постбинарного компьютинга при переходе к расширенному кодо-логическому базису, которое позволило разработать арифметико-логические основы для обеспечения достоверности результатов компьютерных вычислений в системах обработки информации.

Результаты диссертационного исследования могут быть сформулированы следующим образом:

1. Рассмотрены вопросы обеспечения эффективности, средства и методы современного компьютинга. Выполнены анализ и достаточные обоснования для возможности перехода к модификации системы компьютерных вычислений с целью повышения ее надежности и адекватности к современным требованиям путем дальнейшего развития как логической, так и вычислительной составляющей современного компьютинга.

2. Проанализированы основные тенденции перехода к пространству тетралогики. Реализованы нольместные, одноместные и двуместные логические операции. Выполнены представления функций тетралогики с использованием свойств аппроксимационной решетки и аксиоматического аппарата теории множеств.

3. Разработаны основные методы тетракодирования, с применением которых сформирована методика постбинарных вычислений путем выведения основных свойств арифметических операций над тетракодами.

4. Предложены и обоснованы постбинарные форматы чисел с плавающей запятой различных классов точности, в полях которых содержится информация о способе кодирования хранимых числовых данных и их типе (число, значения числителя и знаменателя дроби, значения границ интервала). Для разработанных форматов предложен метод постбинарного округления, позволяющих записать в поле формата интервальное окружение исходного числа.

5. Представлены примеры программной реализации постбинарного кодирования и постбинарных вычислений, демонстрирующей возможности и особенности реализации идей постбинарного компьютинга на базе существующих компьютерных технологий.

СПИСОК ЛИТЕРАТУРЫ

1. Ракитов, А. И. Системный анализ и аналитические исследования: руководство для профессиональных аналитиков : монография / А. И. Ракитов, Д. А. Бондяев, И. Б. Романов [и др.] ; отв. ред. А. И. Ракитов. — Москва, 2009. — 448 с.

2. Чернышов, В. Н. Теория систем и системный анализ : учеб. пособие / В. Н. Чернышов, А. В. Чернышов. — Тамбов : Изд-во Тамбов. гос. техн. ун-та, 2008. — 96 с.

3. Вентцель, Е. С. Исследование операций: задачи, принципы, методология : учеб. пособие для студентов высш. учеб. заведений / Е. С. Вентцель.— 4-е изд., стер. — Москва : Дрофа, 2006 .— 207с.

4. Вентцель, Е. С. Теория вероятностей: учеб. для студентов вузов / Е. С. Вентцель.— 8-е изд., стер. — Москва: Высш. шк., 2002.— 575 с.

5. Антонов, А. В. Системный анализ. Методология. Построение моделей : учеб. пособие по курсу «Системный анализ» /А. В. Антонов . — Обнинск : ИАТЭ, 2001. — 272 с.

6. Антонов, А. В. Системный анализ. Математические модели и методы: учеб. пособие по курсу «Системный анализ» / А. В. Антонов. — Обнинск : ИАТЭ, 2002. — 114 с.

7. Волкова, В. Н. Теория систем: учеб. пособие / В. Н. Волкова, А. А. Денисов. — Москва : Высш. шк., 2006. — 511 с. : ил.

8. Волкова, В. Н. Теория систем и системный анализ : учеб. для акад. бакалавриата / В. Н. Волкова, А. А. Денисов. — 2-е изд., перераб. и доп. — Москва : Юрайт, 2015. — 616 с. — (Серия «Бакалавр. Академический курс»).

9. Садовский, В. Н. Системный подход и общая теория систем: статус, основные проблемы и перспективы развития / В. Н. Садовский. — Москва : Наука, 1980. –280 с.

10. О’Коннор, Дж. Искусство системного мышления: Необходимые знания о системах и творческом подходе к решению проблем = The Art of Systems

Thinking: Essential Skills for Creativity and Problem Solving / Дж. О'Коннор, И. Макдермотт .— Москва, 2018. —253с.

11. Блауберг, И. В. Системный подход в современной науке / И. В. Блауберг, В. Н. Садовский, Э. Г. Юдин // Проблемы методологии системных исследований. — Москва, 1970. — С. 7–48.

12. Блауберг, И. В. Философский принцип системности и системный подход / И. В. Блауберг, В. Н.Садовский, Э. Г. Юдин // Вопросы философии. — 1978. — № 8. — С. 39–52.

13. Громов, Ю.Ю. Системный анализ в информационных технологиях : учеб. пособие / Ю. Ю. Громов, Н. А. Земской, А. В. Лагутин [и др.]. – 2-е изд., стер. – Тамбов : Изд-во Тамб. гос. техн. ун-та, 2007. — 176 с.

14. Заде, Л. А. Понятие лингвистической переменной и его применение к принятию приближенных решений / Л. А. Заде. — Москва : Мир, 1976. –165 с.

15. Zadeh, L. A. Decision Analysis and Fuzzy logic / L. A. Zadeh // International Conference on Fuzzy Sets and Soft Computing in Economics and Finance (FSSCEF 2004). –Saint-Petersburg, 2004. – P. 645-657. `

16. Zadeh, L. A. Uncertainty in Knowledge Base / L. A. Zadeh, R. R. Yager. — Heidelberg : Springer-Verlag, 1991. — P.459-467.

17. Lukasiewicz, J. Untersuchungen über den Aussagenkalkul / J. Lukasiewicz, A. Tarski // Comptes Rendus des Seances de la Societe des Sciences et des Lettres de Varsovie. — 1930. — Vol. III. — P. 1–21. — Англ. пер.: Investigations into the sentential calculus.

18. Бочвар, Д. А. Об одном трехзначном исчислении и его применении к анализу парадоксов классического расширенного функционального исчисления / Д. А. Бочвар // Математический сборник. — 1938. — Т. 4 (46), № 2. — С. 287–308.

19. Kleene, S. C. On a notation for ordinal numbers / S. C. Kleene // The Journal of Symbolic Logic. — 1938. — № 3. — P. 150–155.

20. Клини, С. К. Введение в метаматематику: математическая логика и рекурсивные функции: пер. с англ. / С. К. Клини. — Изд. 2-е., испр. — Москва: URSS, 2009. — 528 с.

21. Belnap, N. D. How computers should think / N. D. Belnap, G. Ryle // Contemporary Aspects of Philosophy. — Stocksfield, 1977. — P. 30–56.

22. Белнап, Н. Как нужно рассуждать компьютеру / Н. Белнап, Т. Стил // Логика вопросов и ответов / Н. Белнап, Т. Стил. — Москва, 1981. — Разд. 2. — С. 46-53.

23. Belnap, N. A useful four-valued logic. Modern uses of multiple-valued logic / N. Belnap, G. Epstein, J. Dunn // Proceedings of the 1975 : International Symposium on Multiple-valued Logic. — [S. l.], 1976. — P.122-152.

24. Горбатов, А. В. Характеризационная теория синтеза функциональных декомпозиций в k-значных логиках / А. В. Горбатов. — Москва, 2000. — 336 с.

25. Зверев, Г. Н. Неклассические объективные логики с информационной семантикой / Г. Н. Зверев // Вестник Уфимского государственного авиационного технического университета. – 2007. –Vol. 9, №2. – P. 3-15.

26. Зверев, Г. Н. Частотная логика — альтернатива классической логике в новых информационных технологиях / Г. Н. Зверев // Информационные технологии. – 1998. – № 11. – С. 2–10.

27. Зверев, Г. Н. Объективные многозначные логики в интеллектуальных системах моделирования и обработки информации / Г. Н. Зверев // Вестник УГАТУ. – 2003. – Т. 4, № 1. – С. 20–34.

28. Аноприенко, А. Я. Обобщенный кодо–логический базис в вычислительном моделировании и представлении знаний: эволюция идеи и перспективы развития / А. Я. Аноприенко // Інформатика, кібернетика та обчислювальна техніка : зб. ст. / редкол.: Є. О. Башков (голов. ред.) [та ін.]. – Донецьк, 2005. – Вип. 93. – С. 289–316.

29. Аноприенко, А. Я. Тетралогика и тетракоды / А. Я. Аноприенко // Сборник трудов факультета вычислительной техники и информатики / Донец. гос. техн. ун-т. – Донецк, 1996. – Вып. 1. – С. 32–43.

30. Аноприенко, А. Я. Археомоделирование: модели и инструменты докомпьютерной эпохи / А. Я. Аноприенко. – Донецк : УНИТЕХ, 2007. – 318 с.

31. Anoprienko, A. Extended logical and numerical basis for computer simulation / A. Anopriyenko, V. Svjatnyi, A. Reuter // Short Papers Proceedings of the 1997 : European Simulation Multiconference ESM'97, Istanbul, June 1–4, 1997. – Istanbul (SCS), 1997. – P. 21–22.

32. Anoprienko, A. Tetralogic and tetracodes: an effective method for information coding / A. Anopriyenko // Artificial Intelligence and Computer Science : 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics. Berlin, August 24–29, 1997 / Wissenschaft und Technik Verlag. – Berlin, 1997. – Vol. 4. - P. 751–754.

33. Калмыков, С. Л. Методы интервального анализа / С. Л. Калмыков, Ю. И. Шокин, З. Х. Юлдашев. — Новосибирск: Наука. 1986. — 224 с.

34. Алефельд, Г. Введение и интервальные вычисления / Г. Алефельд, Ю. Херцбергер . – Москва : Мир, 1987. – 360 с.

35. Hansen, E. Topics in Interval Analysis / E. Hansen. — London: Oxford University Press, 1969. –132 p.

36. Moore, R. E. Interval analysis. Eiiglewood Cliffs / R. E. Moore. – N. J. : Prentic-e-llall, 1966. – 159 p.

37. Moore, R. E. Methods and Applications of Interval Analysis / R. E. Moore. – SIAM : Philadelphia, 1979. – 345 p.

38. Вощинин, А. П. Интервальный анализ данных как альтернатива регрессионному анализу / А. П. Вощинин, А. Ф. Бочков, Г. Р. Сотиров // Заводская лаборатория. —1990. — № 7. — С. 76–81.

39. Шокин, Ю. И. Интервальный анализ / Ю. И. Шокин. — Новосибирск : Наука, 1981. — 111 с.

40. Назаренко, Т. И. Введение в интервальные методы вычислительной математики / Т. И. Назаренко, Л. В. Марченко — Иркутск : Изд-во Иркут. ун-та, 1982. – 320 с.

41. Шарый, С. П. Конечномерный интервальный анализ / С. П. Шарый ; Ин-т вычислительных технологий СО РАН. — Новосибирск: XYZ, 2019. — 633 с.

42. Брадис, В. М. Опыт обоснования некоторых практических правил действий над приближенными числами / В. М. Брадис // Известия Тверского педагогического института. — Тверь, 1927. — Вып. 3. – С. 32–47.

43. Nickel, K. Mathematics for Computer Science / K. Nickel // Proc. Symposium held in Paris, March 16–18. – 1982. – P. 1-153.

44. Яблонский, С. В. Введение в дискретную математику: учеб. пособие для вузов. — 2-е изд., перераб. и доп. / С. В. Яблонский. — Москва : Наука, 1986. — 384 с.

45. Канторович, Л. В. О некоторых новых подходах к вычислительным методам и обработке наблюдений // Сибирский математический журнал. — 1962. — Т. 3, № 5. – С.701-709.

46. Юровицкий, В. М. О компьютерной «вычислительной катастрофе» [Электронный ресурс] / В. М. Юровицкий. — Режим доступа: http://www.yur.ru/science/computer/Comcat.htm. — Загл. с экрана.

47. Юровицкий, В. М. Метрология для всех. Концепция развития метрологии в XXI веке [Электронный ресурс] / В. М. Юровицкий. — Режим доступа: http://yur.ru/testing/Metrology.htm. — Загл. с экрана.

48. Петров, Ю. П. Обеспечение надежности и достоверности компьютерных расчетов / Петров Ю. П. — Санкт-Петербург : БХВ-Петербург, 2012. — 160 с.

49. Петров, Ю. П. Неожиданное в математике и его связь с авариями и катастрофами / Ю. П. Петров, Л. Ю. Петров. — 4-е изд., перераб. и доп — Санкт-Петербург: БХВ-Петербург, 2005. — 240 с.: ил.

50. Литвинов, Г. Л. Вычислительные алгоритмы с произвольной точностью / Г. Л. Литвинов, Е. В. Маслова // Вычислительные технологии. – 2000. –Т. 5, № 2. – С. 68–70.

51. Маничев, В. Б. Методы численного решения обыкновенных дифференциальных уравнений, нелинейных и линейных алгебраических уравнений с гарантированной достоверностью и точностью : (лекции) / В. Б. Маничев ; МГТУ им. Н. Э. Баумана. - Москва, 2010. – 32 с.

52. Hough, D. Applications of the Proposed IEEE 754 Standard for Floating-Point Arithetic [Electronic resource] / D. Hough, // Computer - 1981. – Режим доступа: http://www.ieeexplore.ieee.org. – Загл. с экрана.

53. Достоверные вычисления. Базовые численные методы : пер. с англ. / У. Кулиш, Д. Рац, Р. Хаммер, М. Хокс. – Москва ; Ижевск : Регулярная и хаотическая динамика, 2005. – 496 с.

54. IEEE Standard for Floating-Point Arithmetic (Revision of IEEE Std 754-1985) // IEEE Computer Society. —2008. — P. 70.

55. Rump, S.M. Algorithm for verified inclusions: theory and practice / S. M. Rump // Reliability in Computing. – 1988. –P. 109–126.

56. Cuyt, A. Remarkable Example of Catastrophic Cancellation Unraveled / A. Cuyt, B. Verdonk, S. Becuve // Computing. — 2001.— № 66 — P. 309–320.

57. Аноприенко, А. Я. Пример Румпа в контексте традиционных, интервальных и постбинарных вычислений / В. А. Гранковский, А. Я. Аноприенко, С. В. Иваница // Наукові праці Донецького національного технічного університету. – Донецьк, 2011. – С. 324–343. – (Серия «Проблемы моделирования и автоматизации проектирования динамических систем» (МАП-2011)» ; вип. 9 (179).

58. Kahan, W. How Futile are Mindless Assessments of Roundoff in Floating-Point Computation [Electronic resource] / W. Kahan. — Режим доступа: https://www.cs.berkeley.edu/~wkahan/Mindless.pdf, 2006. - Загл. с экрана.

59. Arnold, N. Some disasters attributable to bad numerical computing. / Douglas N. Arnold [Electronic resource] / N. Arnold. — Режим доступа: http://www.ima.umn.edu/~arnold/disasters/disasters.html. — Загл.с экрана.

60. Hammer, R. Numerical Toolbox for Verified Computing I : Basic Numerical Problems Theory, Algorithms, and Pascal-XSC Program / R. Hammer, M. Hocks, U. Kulisch, D. Ratz // Springer Publishing Company.- Berlin Heidelberg, — 2012. — P. 354.

61. Литвинов, Г. Л. Универсальные численные алгоритмы и их программная реализация / Г. Л. Литвинов, Е. В. Маслова // International Sophus Lie Centre. — 2000. — С. 16.

62. Litvinov, G. L. Idempotent Mathematics and Interval Analysis / G. L. Litvinov, V. P. Maslov, A. N. Sobolevskii // The Erwin Schroedinger International Institute for Mathematical Physics. – Vienna, 1998. – P. 16.

63. Intel AVX: New frontiers in performance improvements and energy efficiency / N Firasta [at al.] // Intel white paper. — 2008. — P. 9.

64. Lomont, C. Introduction to Intel Advanced Vector Extensions [Electronic resource] / C. Lomont. — Режим доступа: http://software.intel.com/en-us/articles/introduction-to-intel-advanced-vector-extensions. — Загл. с экрана.

65. Seiler, L. Intel MIC Larrabee: A Many-Core x 86 Architecture for Visual Computing / L. Seiler, D. Carmean, E. Sprangle // ACM Transactions on Graphics. — 2008. — Vol. 27, № 3. — P. 18:1–18:15.

66. Аноприенко, А. Я. Тетралогика, тетравычисления и ноокомпьютинг. Исследования 2010–2012 гг. / А. Я. Аноприенгко, С. В. Иваница. – Донецк : ДонНТУ : УНИТЕХ, 2012. – 308 с.

67. Кнут, Д. Искусство программирования. Т. 2 : Получисленные алгоритмы / Д. Кнут. – 3-е изд. — Москва : Вильямс, 2007. — 832 с.

68. Аноприенко, А. Я. Постбинарный компьютинг и интервальные вычисления в контексте кодо-логической эволюции / А. Я. Аноприенко, С. В. Иваница . – Донецк : УНИТЕХ, 2011. – 248 с.

69. Аноприенко, А. Я. Расширенный кодо–логический базис компьютерного моделирования / А. Я. Аноприенко // Информатика, кибернетика и вычислительная техника : сб. науч. тр. Донец. гос. техн. ун–та. – Донецк, 1997. – Вып. 1. – С. 59–64.

70. Нариньяни, А. С. Недоопределенность в системе представления и обработки знаний // Известия АН СССР. Техническая кибернетика. – 1986. – № 5. – С. 3–28.

71. Young, R. C. Algebra of many-valued quantities / R. C. Young // Mathematische Annalen. – 1931. – P. 260–290.

72. Dwyer, P. S. Linear Computations / P. S. Dwyer. — New York : John Wiley & Sons, 1951. — 36 p.

73. Warmus, M. Calculus of approximations / M. Warmus // Bull. Acad. Polon. Sci. –1956. – Vol. IV, № 5. – P. 241-245.

74. Sunaga, T. Theory of an interval algebra and its application to numerical analysis / T. Sunaga // RAAG Memoirs. — 1958. – Vol. 2, misc. II. – P.125-143.

75. Markov, S. The contribution of T. Sunaga to interval analysis and reliable computing / S. Markov, K. Okumura, T. Cendes // Developments in Reliable Computing. – Dordrech : Kluwer Academic Publishers, 1998. – P. 163-184.

76. Клатте, Р. PASCAL-XSC. Язык численного программирования / Р. Клатте, У. Кулиш, М. Неага и др. – Москва: ДМК Пресс, 2000. – 352 с.

77. Агеев, М. П. Алгоритм 61б. Процедуры интервальной математики / М.П. Агеев, В. П. Алик, Ю.И. Марков // Библиотека алгоритмов 51б-100б. — М.: Сов. радио, 1976. — С. 21-26.

78. Hayes, B. A. Lucid Interval / B. A. Hayes // Sigma Xi. –2003. – Vol. 91, № 6. – P. 484–488.

79. Аноприенко, А. Я. Анализ времени выполнения арифметических операций над интервальными числами в СКА Mathematica / А. Я. Аноприенко, С. В. Иваница, Аль Рабаба Хамза // Наукові праці Донецького національного технічного університету. – Донецьк, 2012. – С. 110–115. – (Серія «Інформатика, кібернетика та обчислювальна техніка» ; вип. 16(204).

80. Аноприенко, А. Я. Введение в постбинарный компьютинг. Арифметико-логические основы и программно-аппаратная реализация / А. Я. Аноприенко, С. В. Иваница. — Донецк : ДОННТУ, 2017. – 308 с.

81. Аноприенко А. Я. О некоторых возможностях расширения логического базиса информатики / А. Я. Аноприенко, А. А. Кухтин // Информатизация в условиях перехода к рынку : тез. докл. междунар. науч.-практ. конф., г. Киев, 05–06 нояб. 1992 г. – Киев, 1992. – С. 30–32.

82. Анопрієнко, О. Тетракоди: новий метод кодування сигналів і зображень / О. Анопрієнко, С. Кривошеєв // Оброблення сигналів і зображень та розпізнавання образів : пр. всеукр. міжнар. конф. – Київ, 1996. – С. 15–17.

83. Аноприенко, А. Я. Тетракоды в кодировании и распознавании образов / А. Я. Аноприенко, С. В. Кривошеев, Т. А. Приходько // Информатика, кибернетика и вычислительная техника : сб. науч. тр. / Донец. гос. техн. ун–т. – Донецк, 1997. – Вып. 1. – С. 99–104.

84. Zadeh, L. A. Soft computing and Fuzzy Logic / L. A. Zadeh // Engineering Journal. –1994. – P. 48-56.

85. Аноприенко, А. Я. Представление интервальных чисел средствами постбинарного кодирования / А. Я. Аноприенко, С. В. Иваница // Искусственный интеллект. – 2012. – № 1. – С. 6–16.

86. Аноприенко, А. Я. Интервальные вычисления и перспективы их развития в контексте кодо–логической эволюции / А. Я. Аноприенко, С. В. Иваница // Наукові праці Донецького національного технічного університету. – Донецьк, 2010. – С. 150–160. – (Серія «Проблеми моделювання та автоматизації проектування динамічних систем» ; вип. 8(168).

87. Аноприенко, А. Я. Программная реализация постбинарного кодирования интервалов / А. Я. Аноприенко, С. В. Иваница, Е. В. Бурлака // Наукові праці Донецького національного технічного університету. – Донецьк, 2011. – С. 74–83. - (Серія «Проблеми моделювання та автоматизації проектування» ; вип. 10(197).

88. Иваница, С. В. Интервальный анализ и его применение в рамках традиционных и постбинарных вычислений / С. В. Иваница, Аль Рабаба Хамза, А. Я. Аноприенко // Інформатика та комп'ютерні технології : зб. пр. VII міжнар. наук.–техн. конф. студентів, аспірантів та молодих науковців, 22–23 листоп. 2011 р., м. Донецьк. – Донецьк, 2011. – Т. 1. – С. 249–256.

89. Иваница, С. В. Методы контроля точности результирующих интервалов при вычислении интервальных полиномиальных функций / С. В. Иваница, А. В. Меркулов // Наукові праці Донецького національного технічного університету. -

Донецьк, 2011. - С. 55-60. - (Серія «Інформатика, кібернетика та обчислювальна техніка» ; вип. 14(188).

90. Иваница, С. В. Особенности вычисления интервальных полиномов с контролем точности результирующих интервалов / С. В. Иваница, А. В. Меркулов // Моделювання та комп’ютерна графіка : IV міжнар. наук.-техн. конф., 5-8 жовт. 2011 р., м. Донецьк / Донец. нац. техн. ун-т. – Донецьк, 2011. –С. 119–126.

91. Аноприенко, А. Я. Представление постбинарных форматов чисел с плавающей запятой в контексте интервальных вычислений / А. Я Аноприенко, С. В. Иваница, С. В. Кулибаба // Наукові праці Донецького національного технічного університету. - Донецьк, 2011. – С. 44-49. – (Серія «Інформатика, кібернетика та обчислювальна техніка» ; вип. 14(188).

92. Аноприенко, А. Я. Особенности представления постбинарных вещественных чисел в контексте интервальных вычислений и развития аппаратного обеспечения средств компьютерного моделирования / А. Я. Аноприенко, С. В. Иваница, С. В. Кулибаба // Моделювання та комп’ютерна графіка : IV міжнар. наук.-техн. конф., 5-8 жовт. 2011 р., м. Донецьк / Донец. нац. техн. ун-т. – Донецьк, 2011. – С. 13–19.

93. Аноприенко, А. Я. Периодическая система развития компьютерных систем и перспективы нанокомпьютеризации / А. Я. Аноприенко // Инновационные перспективы Донбасса : материалы междунар. науч.–практ. конф., г. Донецк, 20–22 мая 2015 г. – Донецк, 2015. – Т. 5 : Компьютерные науки и технологии. – С. 14–22.

94. Аверкин, А. Н. Толковый словарь по искусственному интеллекту / А. Н. Аверкин, М. Г. Гаазе-Рапопорт, Д. А. Поспелов — Москва : Радио и связь, 1992. – 256 с.

95. Лукасевич, Я. О принципе противоречия у Аристотеля. Критическое исследование / Я. Лукасевич ; пер. с польс.; общ. ред., вступ. ст. проф. А. С. Карпенко. – Москва ; Санкт – Петербург : Центр гуманит. инициатив, 2012. – 255 с.

96. Лукасевич, Я. Аристотелевская силлогистика с точки зрения современной формальной логики / Я. Лукасевич ; пер. с англ. Н. И. Стяжкина, А. Л. Субботина; общ. ред. и вступ. ст. П. С. Попова. — Москва: Тривиум, 2000. – 316 с.

97. Иваница, С. В. Арифметические основы вычислительных систем. Арифметика чисел с фиксированной запятой : учеб. пособие для вузов / С. В. Иваница ; ГОУВПО "ДОННТУ". — Донецк : УНИТЕХ, 2018. – 360 с.

98. Малая цифровая вычислительная машина «Сетунь» / Н. П. Брусенцов, С. П. Маслов, В. П. Розини [др.]. – Москва: Изд-во Моск. ун-та, 1962. – 140 с.

99. Брусенцов Н. П. Искусство достоверного рассуждения. Неформальная реконструкция аристотелевой силлогистики и булевой математики мысли / Н. П. Брусенцов . — Москва: Новое тысячелетие, 1998. – 132 с.

100. Брусенцов, Н. П. Логика и интеллект / Н. П. Брусенцов // Искусственный интеллект. — Донецк, 2004. – № 2. – С. 28–31.

101. Брусенцов, Н. П. Воссоздание аристотелевой безукоризненной силлогистики / Н. П. Брусенцов // Современные информационные технологии и ИТ-образование. – 2010. –Т. 2, № 6. – С. 73–76.

102. Belnap, N. D. A useful four-valued logic. Modern uses of multiple-valued logic / Belnap N. D., Epstein G., Dunn J. M. // Multiple-valued Logic Proceedings : International Symposium. –1975. – P. 5-37.

103. Belnap, N. D. Erotetio logic: an introduction to the logic of questions and answers / N. D. Belnap, T. B. Steel . –Yale University Press, forthcoming. –1976. – P. 11-24.

104. Ивин, А. А. Логика: учеб. пособие / А. А. Ивин. – 2-е изд. — Москва: Знание, 1998. – 228 с.

105. Иваница, С. В. Особенности реализации операций тетралогики / С. В. Иваница, А. Я. Аноприенко // Наукові праці Донецького національного технічного університету. – Донецьк, 2011. – С. 134–140. – (Серія «Інформатика, кібернетика та обчислювальна техніка» ; вип. 13(185).

106. Иваница, С. В. Реализация логических операций над элементами тетракодов / С. В. Иваница, А. Я. Аноприенко // Інформаційні управляючі системи

та комп’ютерний моніторинг (ІУС КМ-2011) : матеріали ІІ Всеукр. наук.-техн. конф. студентів, аспірантів і молодих вчених, 11–13 квіт. 2011 р., м. Донецьк. — Донецьк, 2011. – Т. 2. – С. 198–202.

107. Марченков, С. С. Замкнутые классы булевых функций / С. С. Марченков. – Москва : ФизМатЛит, 2000. –128 с.

108. Weisstein, E. W. Hasse Diagram. From MathWorld [Electronic resource] / E. W. Weisstein. –Режим доступа: http://mathworld.wolfram.com/HasseDiagram.html. – Загл. с экрана.

109. Колмогоров, А. Н. Математическая логика / А. Н. Колмогоров, А. Г. Драгалин. – Изд. 3-е, стер. – Москва : КомКнига, 2006. – 256 с.

110. Simpson, R. E. Introductory Electronics for Scientists and Engineers / R. E. Simpson. – 2-nd ed. –Boston: MA: Allyn and Bacon, 1987. – 540p.

111. Баркалов, А. А. Прикладная теория цифровых автоматов / А. А. Баркалов, Л. А. Титаренко; ГВУЗ «ДонНТУ». – Донецк : УНИТЕХ, 2010. – 320 с.

112. Гиндикин, С. Г. Алгебра логики в задачах / С. Г. Гиндикин. – Москва : Наука, 1972. – 288 с.

113. Аноприенко, А. Я. Развитие многомерной логики / А. Я. Аноприенко, С. Г. Джура, С. В. Иваница // Дельфис. – 2016. – № 4(88). – С. 85-91.

114. Anoprienko, A. Y. Development of multidimensional logic: towards the universal brotherhood [Электронный ресурс] / A. Y. Anoprienko, S. G. Dzhura, S. V. Ivanitsa // Современные проблемы техносферы и подготовки инженерных кадров = Les problèmes contemporains de latechnosphère et de la formation des cadres d’ingénieurs : сб. трудов IX Междунар. науч.-метод. конф., 1–9 окт. 2016 г., г. Сухум / ГОУВПО "ДОННТУ" [и др.]. — Электрон. дан. (1 файл: 11 Мб). – Донецк : ДОННТУ, 2016. — Систем. требования: ZIP-архиватор.

115. Аноприенко, А. Я. Особенности реализации постбинарных логических операций / А. Я. Аноприенко, С. В. Иваница // Искусственный интеллект. –2011. – № 2. – С. 110–121.

116. Hayes, N. T. Trits to Tetrits [Electronic resource] / N. T. Hayes. – Режим доступа: http://grouper.ieee.org/groups/1788/PositionPapers/Tetrits.pdf. – Загл с экрана.

117. Габидулин, Э. М. Лекции по теории информации / Э. М. Габидулин, Н. И. Пилипчук. – Москва : МФТИ, 2007. – 214 с.

118. Anopriyenko, A. Postbinary calculations as a machine – assisted realization of real interval calculations [Электронный ресурс] / A. Anopriyenko, S. Ivanitsa, A. Hamzah, A. Adnan // International Journal of Science and Applied Information Technology (IJSAIT). –Электрон. дан. – 2014. – Т. 3, № 6. – Режим доступа: http://warse.org/ pdfs/2014/ijsait01362014.pdf. – Загл. с экрана.

119. Аноприенко, А. Я. Закономерности развития аналого-цифровых преобразователей и перспективы использования постбинарного кодирования [Электронный ресурс] / А. Я. Аноприенко, Р. Л. Варзар, С. В. Иваница // Наукові праці Донецького національного технічного університету. — Електрон. дані (1 файл: 22 Мб). — Донецьк, 2014. — (Серія «Інформатика, кібернетика та обчислювальна техніка» ; вип. 1(19). — Систем. вимоги: Acrobat Reader.

120. Аноприенко, А. Я. Интервальные вычислительные модули для расчета параметров производительности серверных компьютерных систем [Электронный ресурс] / А. Я. Аноприенко, С. В. Иваница, Хамза Аль-Рабаба // Системний аналіз та інформаційні технології у науках про природу та суспільство: розробки і досягнення кафедри комп'ютерних систем моніторингу : всеукр. зб. наук. пр. / ДВНЗ «ДонНТУ». — Електрон. дані (1 файл: 54 Мб). — Донецьк, 2012. — № 1(2)–2(3). — Систем. вимоги: Acrobat Reader.

121. Аноприенко, А. Я. Особенности аппаратной реализации обобщенного клеточного тетраавтомата [Элетронный ресурс] / А. Я. Аноприенко, Е. Е. Федоров, С. В. Иваница, Хамза Аль-Рабаба // Технологический аудит и резервы производства. Информационно-управляющие системы. – Электрон. дан. – 2015. — № 1/3(21). — Режим доступа: http://cyberleninka.ru/article/n/osobennosti-apparatnoy-realizatsii-obobschennogo-kletochnogo-tetraavtomata. – Загл. с экрана.

122. Аноприенко, А. Я. Постбинарный компьютинг, Grid и «облачные вычисления»: новые реальности компьютерного моделирования [Электронный ресурс] / А. Я. Аноприенко, А. П. Коноплева, Хасан Аль Абабех // Электронный архив Донецкого национального технического университета (г. Донецк). – Электрон. дан. — Донецк, 2006 — 2016. - Режим доступа: http://192.168.253.252:8080/jspui/bitstream/123456789/3405/1/ 2009-10-07-anopriyenko-postbinary-computing-simulation.pdf. – Загл. с экрана.

123. Аноприенко, А. Я. Опыт применения гиперкодов в моделировании клеточных автоматов / А. Я. Аноприенко, А. П. Коноплева // Наукові праці Донецького національного технічного університету. – Донецьк, 2007. – С. 220–227. – (Серія «Проблеми моделювання та автоматизації проектування динамічних систем» ; вип. 6(127).

124. Бурлака, Е. В. Программная модель отображения тетракода на множествах действительных и интервальных чисел / Е. В. Бурлака, С. В. Иваница, А. Я. Аноприенко // Інформатика та комп'ютерні технології : зб. пр. VII міжнар. наук.-техн. конф. студентів, аспірантів та молодих науковців, 22–23 листоп. 2011 р., м. Донецьк / Донец. нац. техн. ун-т. — Донецьк, 2011. — Т. 1. — С. 336–342.

125. Аноприенко, А. Я. Аппаратная реализация преобразователя чисел в постбинарный формат / А. Я. Аноприенко, С. В. Иваница, С. В. Кулибаба // Искусственный интеллект. Интеллектуальные системы ИИ-2012 : материалы Междунар. науч.-техн. конф. (пос. Кацивели, АР Крым, 1–5 окт. 2012 г.). — Донецьк, 2012. — С. 13–16.

126. Анопрієнко О. Я. Принцип роботи, структура і моделювання блоку перетворювача форматів у складі постбінарного співпроцесора / О. Я. Анопрієнко, С. В. Іваниця, С. В. Кулібаба // Інформаційні технології та комп’ютерна інженерія.— Вінниця, 2013. — № 1 (26). - С. 59–65.

127. Аноприенко, А. Я. Особенности постбинарного кодирования на примере интервального представления результатов вычислений по формуле Бэйли-Боруэйна-Плаффа / А. Я. Аноприенко, С. В. Иваница // Наукові праці Донецького

національного технічного університету. — Донецьк, 2010. — С. 19–23. — (Серія «Інформатика, кібернетика та обчислювальна техніка»; вип. 11(164).

128. Аноприенко, А. Я. Реализация интервальных вычислений средствами математического пакета SCILAB с использованием интервального расширения INT4SCI / А. Я. Аноприенко, С. В. Иваница // Донбас-2020: перспективи розвитку очима молодих вчених : матеріали V наук.-практ. конф., м. Донецьк, 25-27 трав. 2010 р. / Донец. обл. держ. адм. [та ін.] ; редкол.: О. А. Мінаєв [та ін.]. — Донецьк, 2010. — С. 629–633.

129. Гранковский, В. А. Подходы к сложным вычислениям в различных математических пакетах / В. А. Гранковский, С. В. Иваница, А. Я. Аноприенко // Информатика и компьютерные технологии : материалы VI междунар. науч.-техн. конф. студентов, аспирантов и молодых ученых, 23-25 нояб. 2010 г. / Донец. нац. техн. ун-т. — Донецк, 2010. — Т. 2. — С. 135–140.

130. Иваница, С. В. Интервальные вычисления в математических пакетах Scilab и Matematica / С. В. Иваница, А. В. Меркулов, А. Я. Аноприенко // Інформатика та комп'ютерні технології : зб. матеріалів VI міжнар. наук.-техн. конф. студентів, аспірантів та молодих науковців, 23–25 листоп. 2010 р., м. Донецьк / ДВНЗ "ДонНТУ". — Донецьк, 2010. — С. 240–246.

131. Bailey, D. H. Plouffe Simon On the Rapid Computation of Various Polylogarithmic Constants / D. H. Bailey, P. B. Borwein // Mathematics of Computation. — 1997. — В. 218, т. 66. — P. 903–913.

132. Скворцов, В. А. Примеры метрических пространств / В. А. Скворцов. – Москва. –2002. — С. 3-24. – (Серия «Библиотека «Математическое просвещение»; вып. 9).

133. Добронец, Б. С. Интервальная математика: учеб. пособие / Б. С. Добронец ; Краснояр. гос. ун-т. — Красноярск, 2004. — 216 с.

134. Иваница, С. В. Реализация арифметических операций сложения и вычитания над тетракодами / С. В. Иваница // Наукові праці Донецького національного технічного університету. — Донецьк, 2013. — С. 149–158. — (Серія «Проблеми моделювання та автоматизації проектування» ; № 1(12)–2(13).

135. Knuth, D. The Art of Computer Programming. Vol. 2: Seminumerical Algorithms / D. Knuth, T. Edition. – Massachusetts: Addison-Wesley, 1997. – 762 p.

136. Бибило, П. Н. Основы языка VHDL: учеб. пособие / П. Н. Бибило. — Изд. 5-е. — Москва : ЛИБРОКОМ, 2012. — 328 с.

137. Таненбаум, Э. Архитектура компьютера. / Э. Таненбаум. — Изд. 6-е. — Санкт-Петербург : Диалектика, 2016. — 816 с.

138. Аноприенко, А. Я. Гибкая разрядность и постбинарные форматы представления вещественных чисел [Электронный ресурс] / А. Я. Аноприенко, С. В. Иваница // Вісник інженерної академії України. — Електрон. дані. — 2012. — № 1. — Режим доступу: http://ea.donntu.org:8080/jspui/bitstream/123456789/14490/1/2012-anopriyenko-ivanitsa-postbinary-formats.pdf. — Назва з екрану.

139. Аноприенко, А. Я. Особенности представления вещественных чисел в постбинарных форматах [Электронный ресурс] / А. Я. Аноприенко, С. В. Иваница // Математичні машини і системи. — Електрон. дані. — 2012. — № 3. — Режим доступу: http://elibrary.ru/item.asp?id=18363419. — Назва з екрану.

140. Аноприенко, А. Я. Программная модель представления чисел в постбинарных форматах [Электронный ресурс] / А. Я. Аноприенко, С. В. Иваница, Е. И. Котов // Искусственный интеллект. Интеллектуальные системы ИИ-2012 : материалы междунар. науч.-техн. конф., АР Крым, пос. Кацивели, 1-5 окт. 2012 г. — Электрон. дан. — Донецк, 2012. — Режим доступа: http://ea.donntu.org:8080/jspui/bitstream/123456789/20026/1/2012-anopriyenko-ivanitsa-kotov-program- model-postbinary-ai-conf.pdf. — Загл. с экрана.

141. Аноприенко, А. Я. Преобразователь вещественных десятичных чисел в постбинарные форматы с плавающей запятой [Электронный ресурс] / А. Я. Аноприенко, С. В. Иваница, Е. И. Котов // Наукові праці Донецького національного технічного університету. — Електрон. дані. — Донецьк, 2012. — (Серія «Проблеми моделювання та автоматизації проектування» ; № 1(10)–2(11). — Режим доступу: http://ea.donntu.org:8080/jspui/bitstream/123456789/17167/1/ 12aajfpf.pdf. - Назва з екрану.

142. Кулибаба, С. В. Программная модель преобразователя вещественных чисел в постбинарные форматы / С. В. Кулибаба, С. В. Иваница // Інформатика та комп'ютерні технології : зб. пр. VII міжнар. наук.-техн. конф. студентів, аспірантів та молодих науковців, 22–23 листоп. 2011 р., м. Донецьк / Донец. нац. техн. ун-т. - Донецьк, 2011. — Т. 1. — С. 214–219.

143. Иваница, С. В. Поддержка динамической разрядности постбинарных форматов чисел с плавающей запятой [Электронный ресурс] / С. В. Иваница, С. В. Кулибаба, А. Я. Аноприенко // Информатика и компьютерные технологии : сб. тр. VIII междунар. науч.-техн. конф. студентов, аспирантов и молодых ученых, г. Донецк, 18–19 сент. 2012 г. — Электрон. дан. — Донецк, 2012. — Т. 2. — Режим доступа: http://ea.donntu.org:8080/jspui/bitstream/123456789/15422/1/5_Кулибаба.pdf. — Загл. с экрана.

144. Иваница, С. В. Преобразователь вещественных чисел с регулируемой точностью / С. В. Иваница, С. В. Кулибаба, А. Я Аноприенко // Информационные управляющие системы и компьютерный мониторинг-2012 (ІУС КМ-2012) : сб. материалов III Всеукр. науч.-техн. конф. студентов, аспирантов и молодых ученых, 16–18 апр. 2012 г., г. Донецк / Донец. нац. техн. ун-т. — Донецк, 2012. — С. 621–624.

145. Анопрієнко, О. Я. Принцип роботи, структура і моделювання блоку перетворювача форматів у складі постбінарного співпроцесора / О. Я. Анопрієнко, С. В. Іваниця, С. В. Кулібаба [Електронний ресурс] // Інформаційні технології та комп’ютерна інженерія. — Електрон. дані (1 файл: 4 Мб). — 2013. — № 1(26). — Систем. вимоги: Acrobat Reader.

146. Anopriyenko, A. Software Implementation of Real Number Conversion into Basic Positional Notations with Controlled Accuracy [Электронный ресурс] / A. Anopriyenko, S. Ivanitsa, S. Kulibaba // International Journal of Science and Applied Information Technology (IJSAIT). — Электрон. дан. — 2013. — № 2(4). — Режим доступа: http://warse.org/pdfs/2013/ ijsait03242013.pdf. — Загл. с экрана.

147. Иваница, С. В. Оценка погрешности представления вещественных чисел в постбинарных форматах с плавающей запятой [Электронный ресурс] /

С. В. Иваница // Искусственный интеллект. — Электрон. дан. (1 файл: 26 Мб). — 2012. — № 4. — Систем. требования: Acrobat Reader.

148. Иваница, С. В. Методы округления чисел с плавающей запятой, представленных в постбинарных форматах [Электронный ресурс] / С. В. Иваница // Искусственный интеллект. Интеллектуальные системы ИИ- 2012 : материалы междунар. науч.-техн. конф., АР Крым, пос. Кацивели, 1–5 окт. 2012 г. — Электрон. дан. — Донецк, 2012. — Режим доступа: http://ea.donntu.org:8080/jspui/ bitstream/ 123456789/ 19603/1/ Ivanitsa ИИ-2012.pdf. — Загл. с экрана.

149. Виноградов, О. Л. Математический анализ / О. Л. Виноградов . — Санкт-Петербург: БХВ-Петербург, 2016. — 752 с.

150. Окулов, С. М. Длинная арифметика / С. М. Окулов // Информатика. – 2000. - № 4. – С. 19–23.

151. Алгоритмы компьютерной арифметики / С. М. Окулов, А. В. Лялин, О. А. Пестов [и др.]. — Москва: БИНОМ. Лаборатория знаний, 2015. — 288 с.

152. Бирюков, А. Г. О гарантированной точности решений задач вычислительной математики в арифметике с плавающей запятой и переменной длиной мантиссы / А. Г. Бирюков, А. И. Гриневич // Труды Московского физико-технического института. – 2012. – Vol. 4, №. 3. – С. 171–180.

153. Наумов, Р. В. Неиспользуемые особенности NET / Р. В. Наумов, А. В. Неустроев // Научные исследования. – 2015. – № 1. – С. 18–19.

154. Казаков, В. Н. Препроцессинг в модулярных алгоритмах / В. Н. Казаков // Вестник российских университетов. Математика. – 2006. –№ 4. – С. 555–557.

155. Половко, А. М. Mathematica для студента / А. М. Половко. — Санкт-Петербург : БХВ-Петербург, 2007. — 368 с.: ил.

Приложение А

Графическая интерпретация унарных логических операций тетралогики

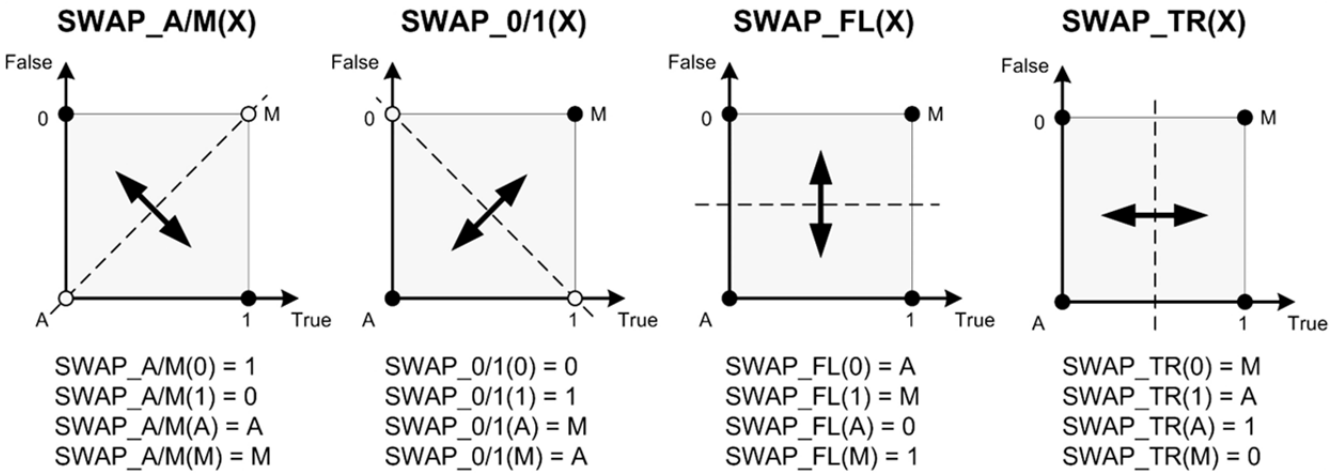


Рисунок А.1 – Графическая интерпретация унарных логических операций инверсной группы (swap group)

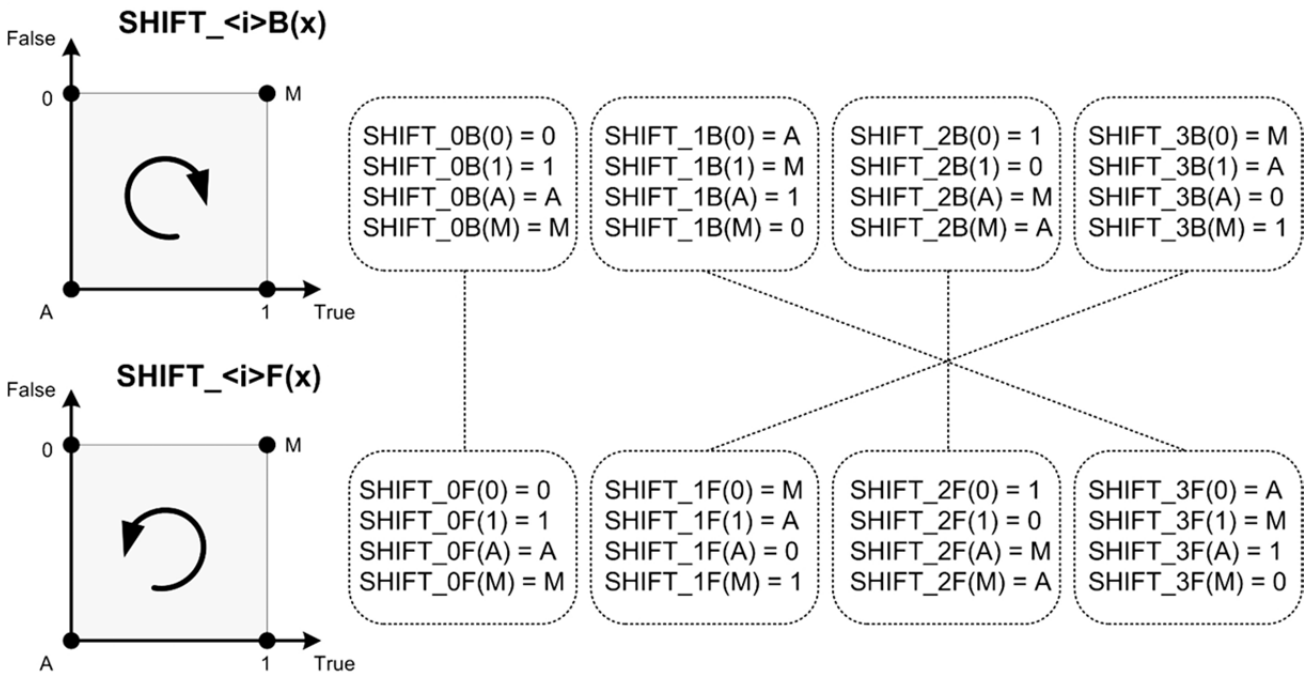


Рисунок А.2 – Графическая интерпретация унарных логических операций сдвиговой группы (shift group)

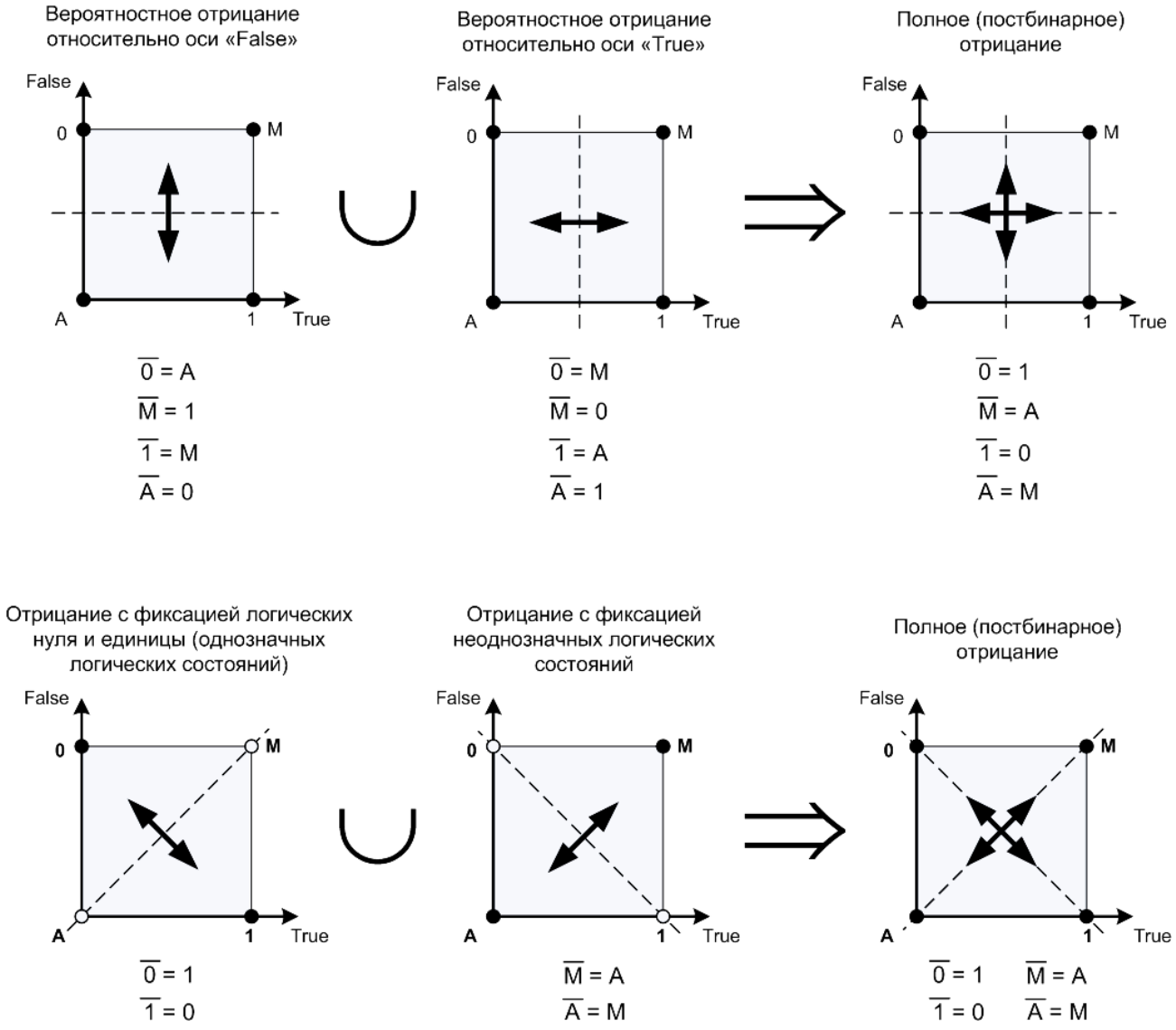


Рисунок А.3 – Графическая интерпретация постбинарного логического отрицания

Приложение Б

Альтернативные способы определения тетрафункций конъюнкции и дизъюнкции

Нахождение результата операций тетралогического сложения и умножения, в которых участвуют значения логических нуля и единицы не вызывает сложности и противоречия, поскольку в таких случаях справедливы свойства аналогичных операций двоичной логики. А результат операций с неопределенностью и множественностью во всех их сочетаниях не столь очевиден, поскольку приводит к противоречию и логическому парадоксу. В этом случае целесообразно дополнительно рассмотреть альтернативные способы определения конъюнкции и дизъюнкции в тетралогике для любых сочетаний неопределенности и множественности.

По своим определениям логические состояния А и М могут быть представлены в виде совокупности и системы логических нуля и единицы соответственно (см. рисунок А.2):

$$\mathrm{A} = \begin{bmatrix} 0 \\ 1; \end{bmatrix} \qquad \mathrm{M} = \begin{cases} 0 \\ 1. \end{cases}$$

Учитывая определения совокупности и системы выражений, очевидно получение следующих равенств:

$$\begin{bmatrix} 0 \\ \mathrm{A} \end{bmatrix} = \mathrm{A}; \qquad \begin{bmatrix} 1 \\ \mathrm{A} \end{bmatrix} = 1; \qquad \begin{cases} 0 \\ \mathrm{M} \end{cases} = 0; \qquad \begin{cases} 1 \\ \mathrm{M} \end{cases} = \mathrm{M}.$$

Таким образом, для конъюнкции справедливы следующие соотношения:

$$\mathrm{A} \,\&\, \mathrm{A} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \,\&\, \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \mathrm{A} \end{bmatrix} = \mathrm{A}; \quad \mathrm{M} \,\&\, \mathrm{M} = \begin{cases} 0 \\ 1 \end{cases} \,\&\, \begin{cases} 0 \\ 1 \end{cases} = \begin{cases} 0 \\ 1 \end{cases} = \mathrm{M};$$

$$\mathrm{M} \,\&\, \mathrm{A} = \mathrm{A} \,\&\, \mathrm{M} = \begin{cases} 0 \,\&\, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ 1 \,\&\, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{cases} = \begin{cases} 0 \\ \begin{bmatrix} 0 = \begin{cases} 0 \\ \mathrm{M} \end{cases} = 0. \\ 1 \end{bmatrix} \end{cases}$$

Для дизъюнкции справедливы следующие соотношения:

$$A \vee A = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \vee \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = A; \quad M \vee M = \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \vee \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} = \begin{Bmatrix} M \\ 1 \end{Bmatrix} = M;$$

$$M \vee A = A \vee M = \begin{bmatrix} 0 \vee \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \\ 1 \vee \begin{Bmatrix} 0 \\ 1 \end{Bmatrix} \end{bmatrix} = \begin{bmatrix} A \\ 1 \end{bmatrix} = 1.$$

Результаты поученных выражений соответствуют значениям, приведенным в таблицах 2.2 и 2.3.

Определение конъюнкции и дизъюнкции в тетралогике также возможно с помощью аксиоматического аппарата теории множеств. Современная теория множеств строится на системе аксиом Цермело-Френкеля (ZF – Zermelo-Fraenkel), из которых выводятся все теоремы и утверждения теории множеств. К системе аксиом ZF часто добавляют аксиому выбора и называют системой Цермело-Френкеля с аксиомой выбора (ZFC – Zermelo-Fraenkel set theory with the axiom of Choice).

В контексте тетралогики система ZFC также представляет интерес, поскольку всякое логическое пространство может быть выражено языком теории множеств таким образом, что полученные теоремы станут теоремами о множествах, доказуемыми из аксиом ZFC. Иными словами, поскольку любой объект можно считать множеством, то, соответственно, появляется возможность переформулировать какое-либо утверждение, не нарушив его первоначальную истинность.

Таким образом, пространство тетралогики можно представить совокупностью четырех непустых множеств: *Q*, *J* – содержащих по одному фиксированному логическому элементу (значение «Истина» и значение «Ложь»); *A*, *M* – содержащих, согласно определению логических состояний «Множественность» и «Неопределенность», по два элемента, представляющих собой результат логической операции над элементами множеств *Q*, *J*. Математическая запись определения множеств *Q*, *J*, *A* и *M* [106]:

$$\forall q\, \exists Q \left(q \subseteq Q \to q = 0 \right); \tag{Б.1}$$

$$\forall j\, \exists J \left(j \subseteq J \to j = 1 \right); \tag{Б.2}$$

$$\forall a\, \exists A \left(a \subset A \to a \in Q \vee a \in J \right); \tag{Б.3}$$

$$\forall m\, \exists M \left(m \subset M \to m \in Q \wedge m \in J \right). \tag{Б.4}$$

Из записей (Б.1) и (Б.2) очевидно, что любой элемент q и j множеств Q и J соответственно представляет фиксированное логическое значение. При этом $Q = \{0\}$ и $J = \{1\}$, что соответствует определению логических нуля и единицы. Запись (Б.3) показывает, что любой элемент a множества A принадлежит множеству $Q \cup J$ и словесно может быть описан как «или ложь или истина», что соответствует понятию неопределенности в тетралогике. Запись (Б.4) показывает, что любой элемент m множества M принадлежит множеству $Q \cap J$ и словесно может быть описан как «и ложь и истина», что соответствует понятию множественности в тетралогике.

Однако для полного представления пространства тетралогики необходимо ввести еще одно множество-универсум U, представляющее собой пространство, содержащее все четыре ранее объявленных множества. Таким образом,

$$\forall q\ \forall j\ \forall a\ \forall m\ \exists U = \{u \mid u \in q \vee u \in j \vee u \in a \vee u \in m\}. \tag{Б.5}$$

Из записи (Б.5) следует, каждый элемент u множества U представляет собой одно из состояний тетралогики, которое способно каждый элемент множеств Q, J, A и M при определенных обстоятельствах представлять в виде логических нуля или единицы. Данное условие является необходимым и достаточным для того, чтобы получить на универсуме тетралогики результаты операций конъюнкции и дизъюнкции.

Таким образом, с учетом определений четырех состояний тетралогики (Б.1–Б.5) можно сделать следующие заключения:

1) $0 = Q$, в частности, $q \in \{0\}$;

2) $1 = J$, в частности, $j \in \{1\}$);

3) $(A \cap M) \subset (A \cup M) \subset U -$ для множеств A и M на универсуме U определены операции объединения и пересечения;

4) $a \in (Q \cup J)$, $q \subset A$, $j \subset A$, $(Q \cup J) \neq \varnothing$, откуда $a \notin \varnothing \vee a \in \{0,1\}$ – формирование множества для состояния неопределенности A;

5) $m \in (Q \cap J)$, $q \subset M$, $j \subset M$, $(Q \cap J) \neq \varnothing$, откуда $m \notin \varnothing \vee m \in \{0\} \wedge m \in \{1\}$ – выделение противоречия при формировании множества для состояния множественности M: множество M не может быть пустым, однако для множеств Q и J не определена операция пересечения.

Возникшее противоречие в данном представлении является достаточно актуальным, так как позволяет объяснить ряд существующих спорных моментов. Во-первых, данное противоречие проявляется и в определении самой «множественности», поскольку при одновременном поступлении противоречивых высказываний попытка зафиксировать сразу два состояния «Истина» и «Ложь» приведет к логическому парадоксу, при котором современный компьютер (являющийся по своей сути классическим двузначным логическим устройством) должен полностью отказаться от принятия и дальнейшей обработки противоречивой информации. Во-вторых, из двух состояний тетралогики A и M, данное противоречие позволяет в ряде случаев выделить множественность M как доминирующее над неопределенностью A состояние и в качестве альтернативного выхода предложить поочередную подстановку логических «1» и «0» в контексте традиционного бинарного компьютинга.

После того, как состояния тетралогики представлены в виде множеств сохраняя при этом свои логические свойства, применяя аксиомы теории множеств можно реализовать любую логическую операцию над множествами. Так, например, операция отрицания тетралогики может быть получена следующими соотношениями:

$$\forall q \in Q, U \;\; \exists \overline{Q} \;\; \left(\overline{Q} = \left\{\overline{q} : \overline{q} \notin Q \vee \overline{q} \in U \rightarrow \overline{q} \in J\right\}\right);$$

$$\forall j \in J, U \;\; \exists \overline{J} \;\; \left(\overline{J} = \left\{\overline{j} : \overline{j} \notin J \vee \overline{j} \in U \rightarrow \overline{j} \in Q\right\}\right);$$

$$\forall a \in A, U \;\; \exists \overline{A}$$
$$\left(\overline{A} = \left\{\overline{a} : \overline{a} \notin A \vee \overline{a} \in U \to \overline{a} \notin (Q \cup J) \vee \overline{a} \in \left(\overline{Q} \cup \overline{J}\right) \to \overline{a} \in (Q \cap J) = M\right\}\right);$$

$$\forall m \in M, U \;\; \exists \overline{M}$$
$$\left(\overline{M} = \left\{\overline{m} : \overline{m} \notin M \vee \overline{m} \in U \to \overline{m} \notin (Q \cap J) \vee \overline{m} \in \left(\overline{Q} \cap \overline{J}\right) \to \overline{m} \in (Q \cup J) = A\right\}\right).$$

Действительно, соотношения множеств $\overline{Q} = J$, $\overline{J} = Q$, $\overline{A} = M$ и $\overline{M} = A$ эквивалентны соотношениям логических состояний $\overline{0} = 1$, $\overline{1} = 0$, $\overline{\mathrm{A}} = \mathrm{M}$ и $\overline{\mathrm{M}} = \mathrm{A}$, что соответствует выполнению рассмотренной ранее тетрафункции отрицания.

Для обозначения логического сложения необходимо принять следующие обозначения: если φ_1 – элемент множества Φ_1, а φ_2 – элемент множества Φ_2, причем каждое из множеств Φ_1 и Φ_2 является подмножеством универсума U, то справедлива следующая запись (с точки зрения теории множеств, дизъюнкция аналогична операции объединения):

$$(\varphi_1 \vee \varphi_2) \in (\boldsymbol{\Phi}_1 \cup \boldsymbol{\Phi}_2), \;\; \boldsymbol{\Phi}_1, \boldsymbol{\Phi}_2 \subset U.$$

С учетом того, что операция объединения $\boldsymbol{\Phi}_1 \cup \boldsymbol{\Phi}_2$ может быть замещена множеством $\boldsymbol{\Phi} = \{\varphi \mid \varphi \in \boldsymbol{\Phi}_1 \vee \varphi \in \boldsymbol{\Phi}_2\}$, для множеств Q и J справедливы следующие равенства:

$$Q \cup Q = Q; \text{ и } J \cup J = J.$$

Аналогично сложению в двоичной логике, высказывание $\varphi_1 \vee \varphi_2$ будет истинно тогда и только тогда, когда хотя бы одно высказывание φ_1 или φ_2 истинно. Так для множеств Q и J операция дизъюнкции определена следующим образом:

$$Q \vee Q = Q = \{0\}; \;\; Q \vee J = J = \{1\}; \;\; J \vee J = J = \{1\}.$$

В записи (2.9) множество A определено как множество, состоящее из элементов множеств Q или J. Поэтому для элементов множеств Q, J и A операция дизъюнкции определяется следующим образом:

$$A \vee Q = A; \qquad A \vee J = J.$$

Операция дизъюнкции двух множеств A, в силу представления их элементов как неопределенностей значений «Истина» или «Ложь», отображена в самом множестве A (эквивалентно высказыванию «неопределенность или неопределенность есть неопределенность»):

$$A \vee A = A.$$

В записи (Б4) множество M определено как множество элементов, одновременно принадлежащих множествам Q и J. Поэтому для множеств Q, J и M справедливы следующие отношения дизъюнкции:

$$M \vee Q = M; \quad M \vee J = J.$$

Операция дизъюнкции двух множеств M, в силу их представления как многозначности, т. е. одновременности появления истины и лжи, также отображена в самом множестве M (эквивалентно высказыванию «многозначность или многозначность есть многозначность»):

$$M \vee M = M.$$

Операция дизъюнкции множеств A и M описывается следующим соотношением:

$$\forall \varphi_1 \in A,U \;\; \forall \varphi_2 \in M,U$$
$$\exists \Phi = \Phi_1 \vee \Phi_2 \;\; \left(\Phi = \left\{\varphi : (\varphi \in Q \vee \varphi \in J) \vee (\varphi \in Q \wedge \varphi \in J) \rightarrow \varphi \in J\right\}\right).$$

Таким образом, соотношения множеств (Таблица Б.1) эквивалентны идентичным соотношениям логических состояний, что соответствует выполнению рассмотренной ранее тетрафункции дизъюнкции.

Для обозначения логического умножения также принимаются следующие обозначения: если φ_1 – элемент множества Φ_1, а φ_2 – элемент множества Φ_2, причем каждое из множеств Φ_1 и Φ_2 является подмножеством универсума U, то справедлива следующая запись (с точки зрения теории множеств, конъюнкция аналогична операции пересечения):

$$\left(\varphi_1 \wedge \varphi_2\right) \in \left(\Phi_1 \cap \Phi_2\right), \;\; \Phi_1, \Phi_2 \subset U.$$

Так как операция объединения $\Phi_1 \cap \Phi_2$ может быть замещена множеством $\Phi = \{\varphi \mid \varphi \in \Phi_1 \wedge \varphi \in \Phi_2\}$, то для множеств Q и J справедливы следующие равенства: $Q \cap Q = Q$; и $J \cap J = J$.

Как и в двоичной логике, высказывание $\varphi_1 \wedge \varphi_2$ будет истинно тогда и только тогда, когда оба высказывания φ_1 и φ_2 истинны. Так для множеств Q и J операция дизъюнкции определена следующим образом:

$$Q \wedge Q = Q = \{0\}; \quad Q \wedge J = Q = \{0\}; \quad J \wedge J = J = \{1\}.$$

В записи (2.8) множество A определено как множество, состоящее из элементов множеств Q или J. Поэтому справедливы следующие отношения конъюнкции

$$A \wedge Q = Q; \qquad A \wedge J = A.$$

Аналогично дизъюнкции, конъюнкция двух множеств A отображена в самом множестве A (эквивалентно высказыванию «неопределенность и неопределенность есть неопределенность»):

$$A \wedge A = A.$$

В записи (Б.4) множество M определено как множество элементов, одновременно принадлежащих множествам Q и J. Поэтому справедливы следующие отношения конъюнкции: $M \wedge Q = Q; \qquad M \wedge J = M.$

Операция конъюнкции двух множеств M также отображена в самом множестве M (эквивалентно высказыванию «многозначность и многозначность есть многозначность»):

$$M \wedge M = M.$$

И, наконец, операция дизъюнкции множеств A и M описывается следующим соотношением:

$$\forall \varphi_1 \in A,U \quad \forall \varphi_2 \in M,U$$
$$\exists \Phi = \Phi_1 \wedge \Phi_2 \quad \left(\Phi = \{\varphi : (\varphi \in Q \vee \varphi \in J) \wedge (\varphi \in Q \wedge \varphi \in J) \rightarrow \varphi \in Q\}\right).$$

Таким образом, соотношения множеств (таблица Б.2) эквивалентны идентичным соотношениям логических состояний, что соответствует выполнению рассмотренной ранее тетрафункции конъюнкции.

Таблица Б.1 – Соотношение выполнения дизъюнкции над множествами и состояниями тетралогики

Дизъюнкция (объединение) множеств	Дизъюнкция состояний тетралогики
$Q \vee Q = Q$ $Q \vee J = J$ $J \vee J = J$ $A \vee Q = A$ $A \vee J = J$ $A \vee A = A$ $A \vee M = J$ $M \vee M = M$ $M \vee Q = M$ $M \vee J = J$	$0 \vee 0 = 0$ $0 \vee 1 = 1$ $1 \vee 1 = 1$ $\mathrm{A} \vee 0 = \mathrm{A}$ $\mathrm{A} \vee 1 = 1$ $\mathrm{A} \vee \mathrm{A} = \mathrm{A}$ $\mathrm{A} \vee \mathrm{M} = 1$ $\mathrm{M} \vee \mathrm{M} = \mathrm{M}$ $\mathrm{M} \vee 0 = \mathrm{M}$ $\mathrm{M} \vee 1 = 1$

Таблица Б.2 – Соотношение выполнения конъюнкции над множествами и состояниями тетралогики

Конъюнкция (пересечение) множеств	Конъюнкция состояний тетралогики
$Q \wedge Q = Q$ $Q \wedge J = Q$ $J \wedge J = J$ $A \wedge Q = Q$ $A \wedge J = A$ $A \wedge A = A$ $A \wedge M = Q$ $M \wedge M = M$ $M \wedge Q = Q$ $M \wedge J = M$	$0 \wedge 0 = 0$ $0 \wedge 1 = 0$ $1 \wedge 1 = 1$ $\mathrm{A} \wedge 0 = 0$ $\mathrm{A} \wedge 1 = \mathrm{A}$ $\mathrm{A} \wedge \mathrm{A} = \mathrm{A}$ $\mathrm{A} \wedge \mathrm{M} = 0$ $\mathrm{M} \wedge \mathrm{M} = \mathrm{M}$ $\mathrm{M} \wedge 0 = 0$ $\mathrm{M} \wedge 1 = \mathrm{M}$

Приложение В

Графическая интерпретация приведения 8-разрядного тетракода к одному или совокупности двоичных кодов

Графическая интерпретация приведения 8-разрядного тетракода к одному или совокупности двоичных кодов размерности [7:0], по аналогии с полями постбинарного клеточного автомата, представлена на рисунке В.1. При этом каждая ячейка определяет одно из чисел совокупности $0 \div (2^8 - 1)$ и пребывает в одном из трех состояний:

1) ☐ – пустая ячейка: однозначно нет числа;
2) ■ – заполненная ячейка: однозначно есть число;
3) ▩ – вероятностная ячейка: число может быть или не быть с равной вероятностью.

На рисунке В.1, *а* тетракод MMMMMMMM приводится ко всем $2^8 = 256$ значениям, а на рисунке В.1, *б* – только к указанным $2^6 = 64$ значениям. Таким образом, можно утверждать, что при приведении тетракода количество двоичных кодов зависит от количества в тетракоде значений М.

На рисунке В1, *в-г* показано, как чередование 0, 1 и М позволяет осуществить выборку совокупности из требуемых значений.

В частности, с помощью 8-разрядного тетракода MMMMMMM0 можно выделить все четные (а с помощью MMMMMMM1 – все нечетные) значения в диапазоне 0..255.

На рисунке В.1, *д* тетракод AAAAAAAA приводится к одному из всех возможных $2^8 = 256$ значений, а на рисунке В1, *е* – к одному из указанных (вероятностных) $2^6 = 64$ значений. Таким образом, можно утверждать, что при приведении тетракода количество двоичных кодов не зависит от количества в тетракоде значений А, однако они определяют область вероятной выборки.

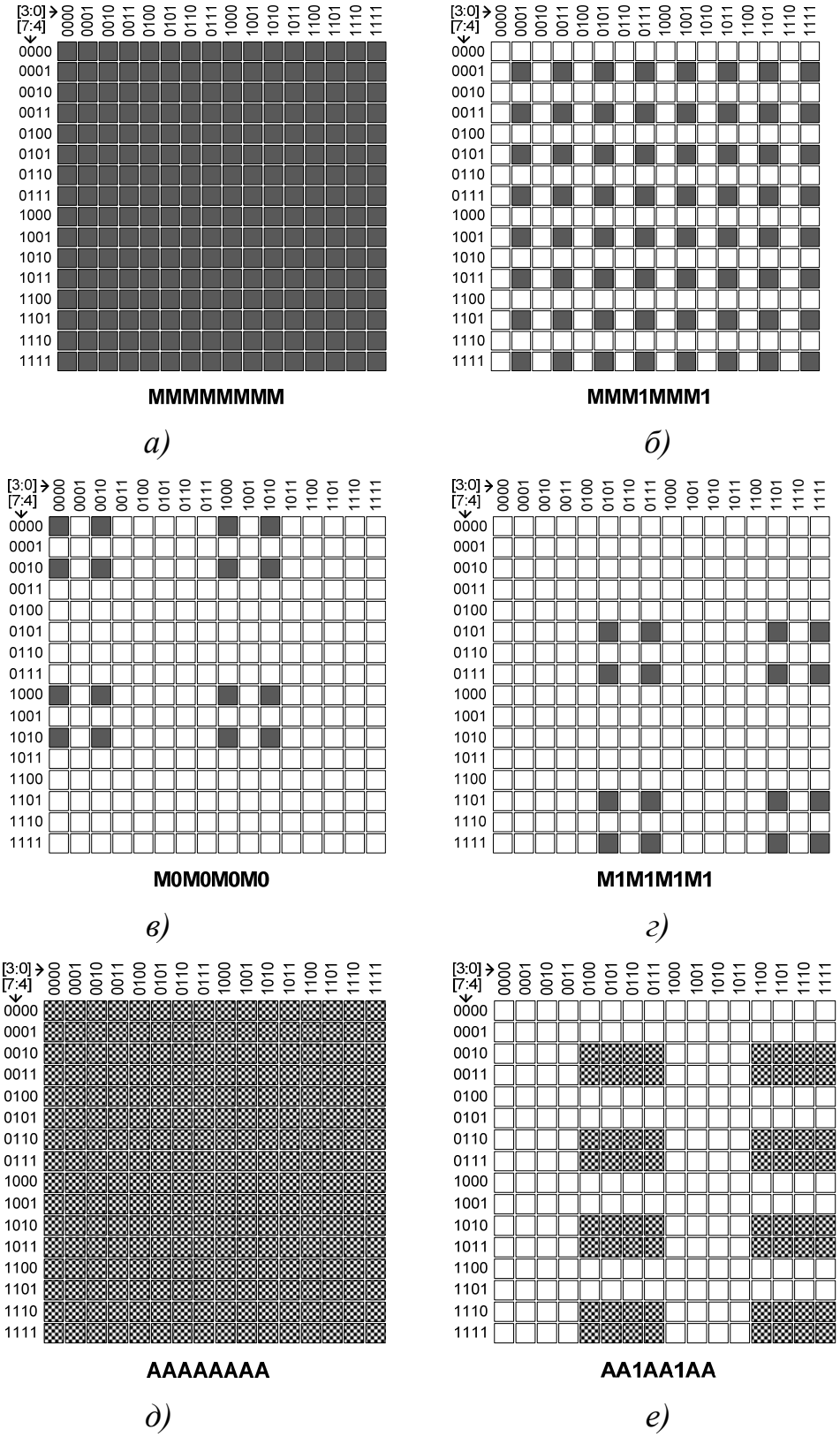
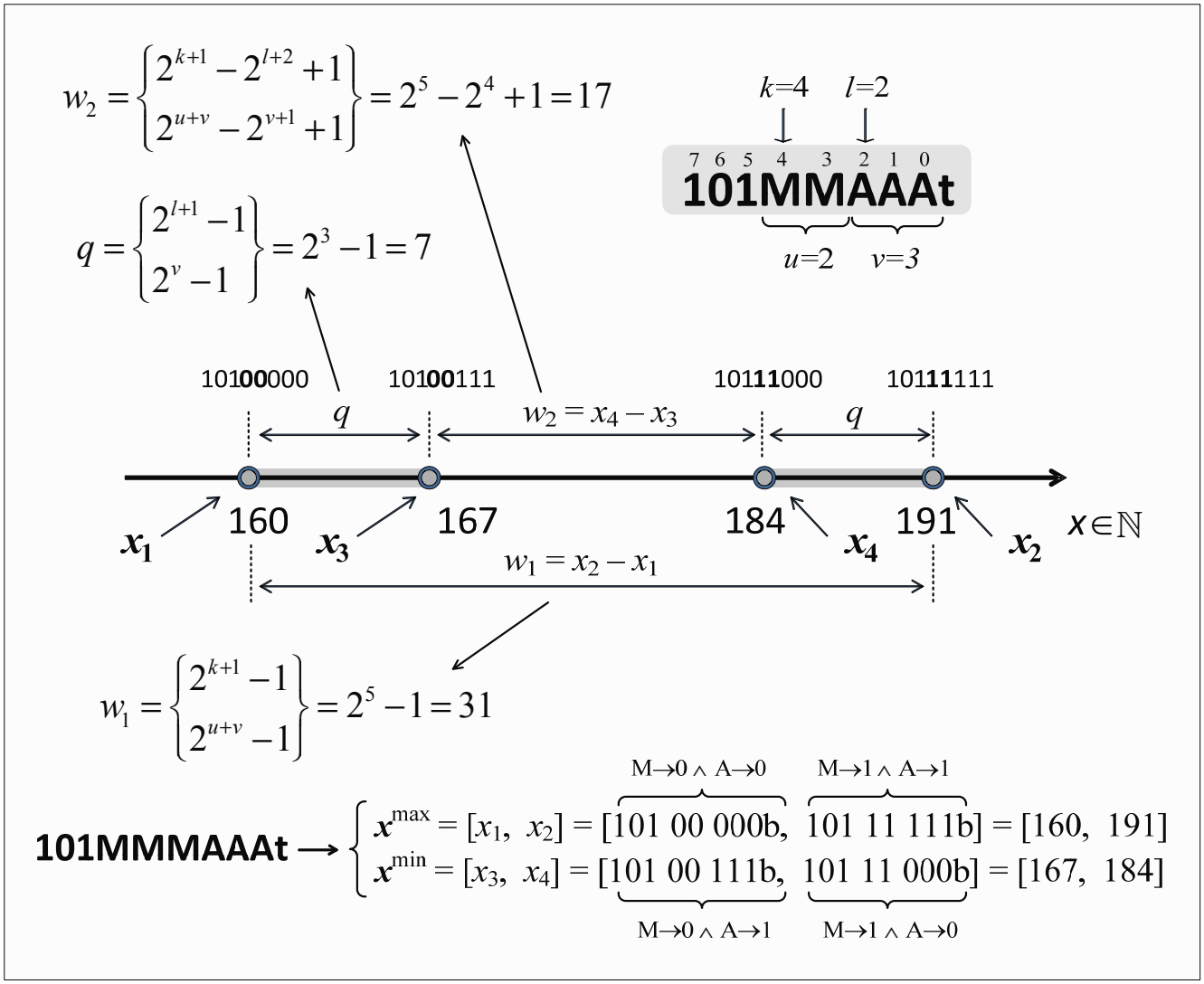


Рисунок В.1 – Графическая интерпретация приведения 8-разрядного тетракода к одному или совокупности двоичных кодов (*а* – тетракод из разрядов М; *б* – чередование групп 3М+1; *в* – чередованание групп М+0; *г* – чередованание групп М+1; *д* – тетракод из разрядов А; *е* – чередование групп 2А+1

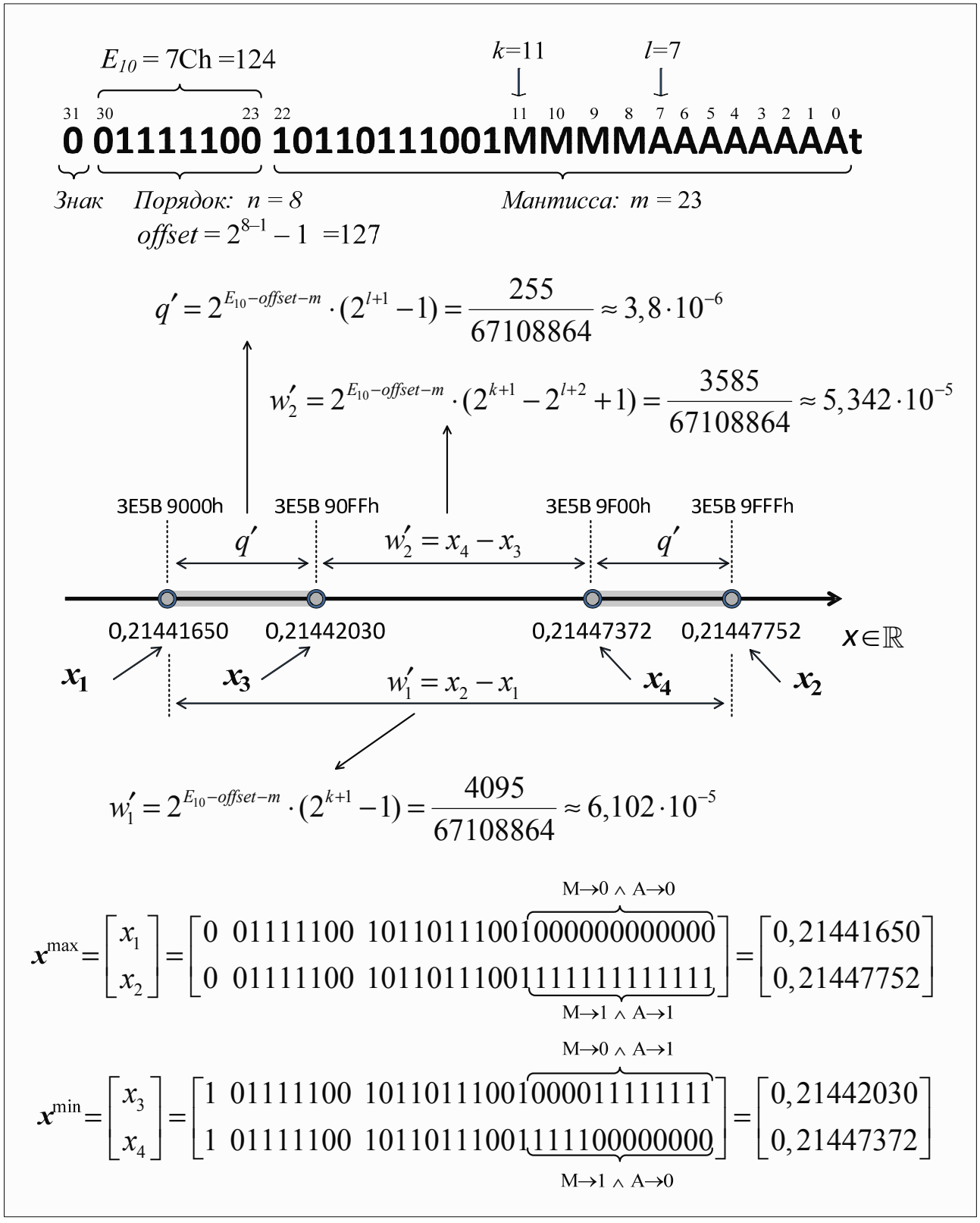
Приложение Г

Примеры расчета предельных значений границ и диапазона их возможного изменения для целочисленного и вещественного интервалов

Получение границ интервала по целочисленному тетракоду 101MMAAA:



Получение границ интервала по 32-разрядному вещественному тетракоду, представленному в формате числа с плавающей запятой:



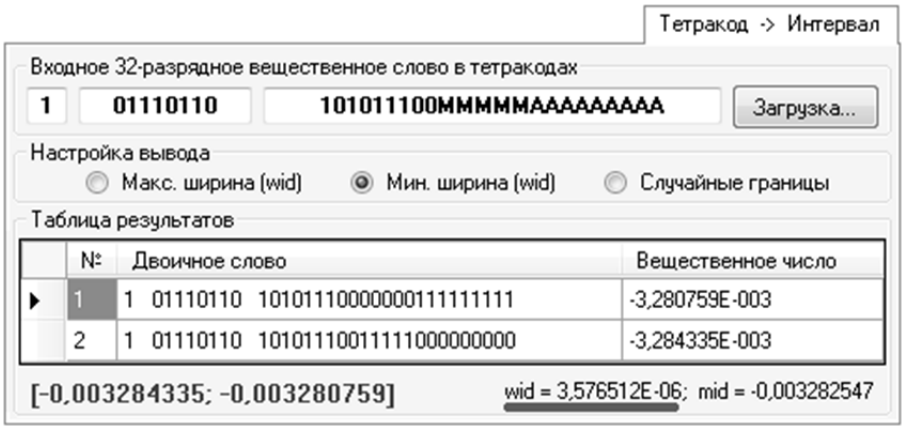
Приложение Д

Результаты приведения тетракода к вещественным интервалам

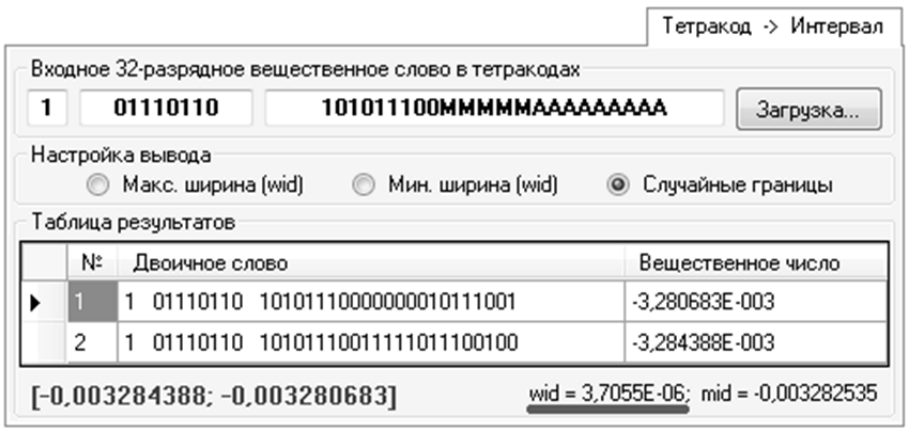
Полученные результаты приведения тетракода к вещественным интервалам, границы которых заданы в формате числа с плавающей запятой одинарной точности с использованием программы «Перевод тетракода в двоичные и десятичные форматы чисел»:

а)

б)

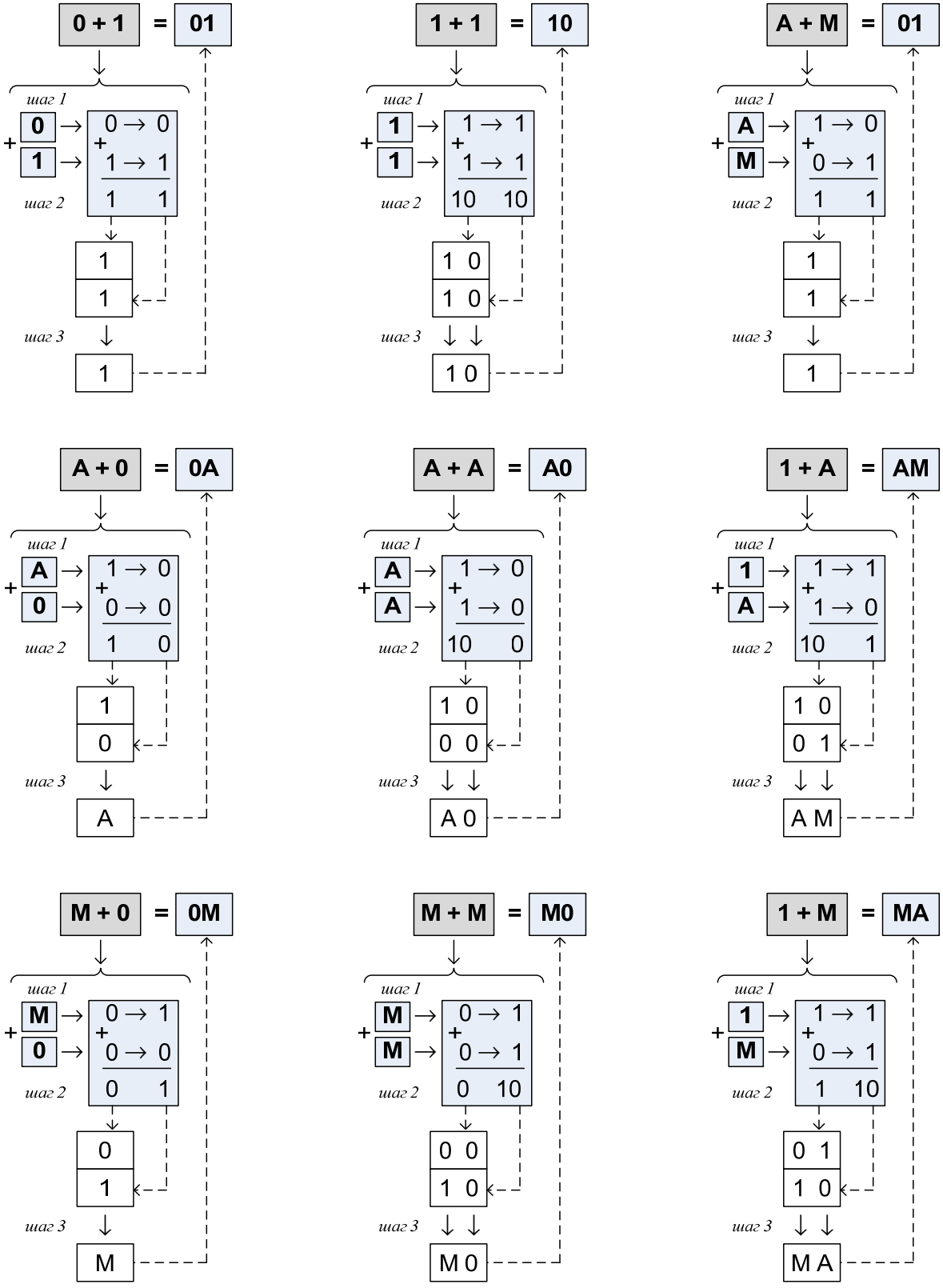


в)



Приложение Е

Схематическая реализация алгоритма выполнения операции сложения двух тетритов



Приложение Ж

Примеры выполнения арифметических операций над тетракодами

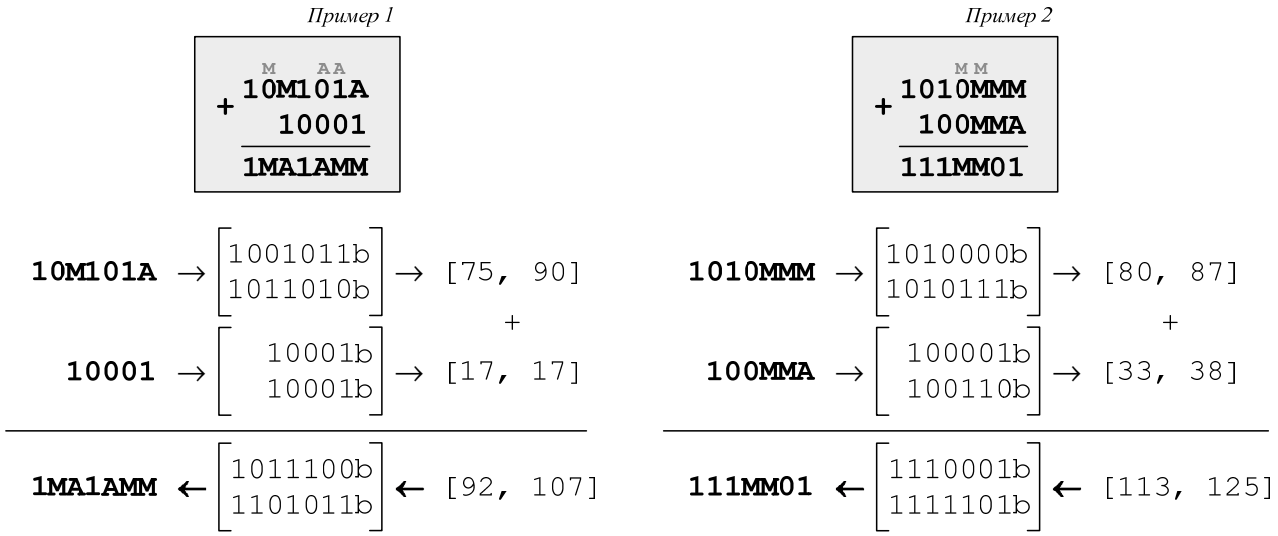


Рисунок Ж.1 – Примеры сложения тетракодов с демонстрацией проверки правильности полученной суммы

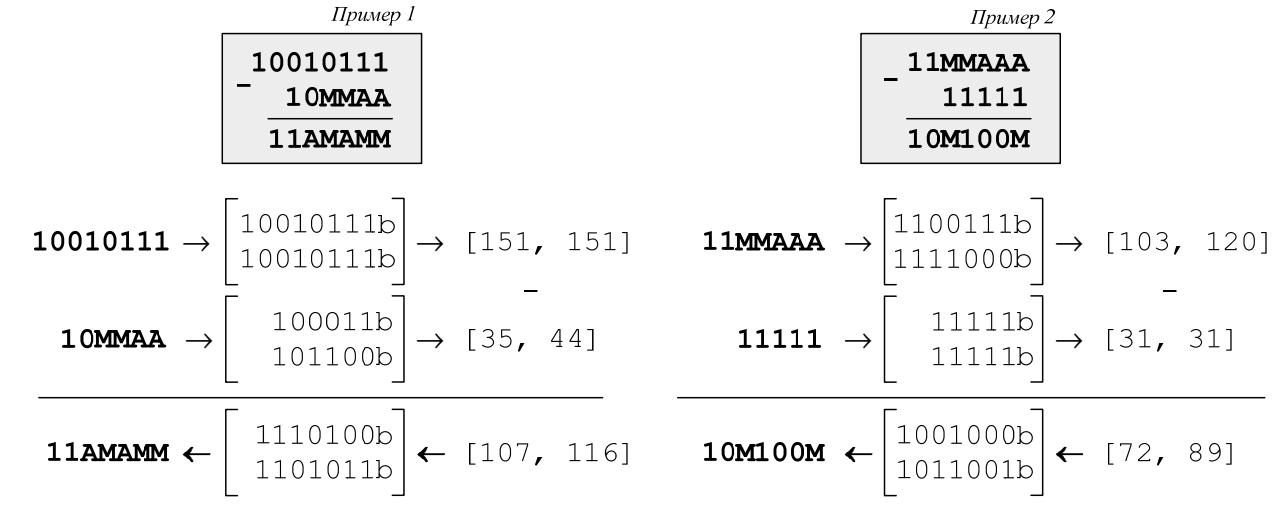


Рисунок Ж.2 – Примеры вычитания тетракодов с демонстрацией проверки правильности полученной разности

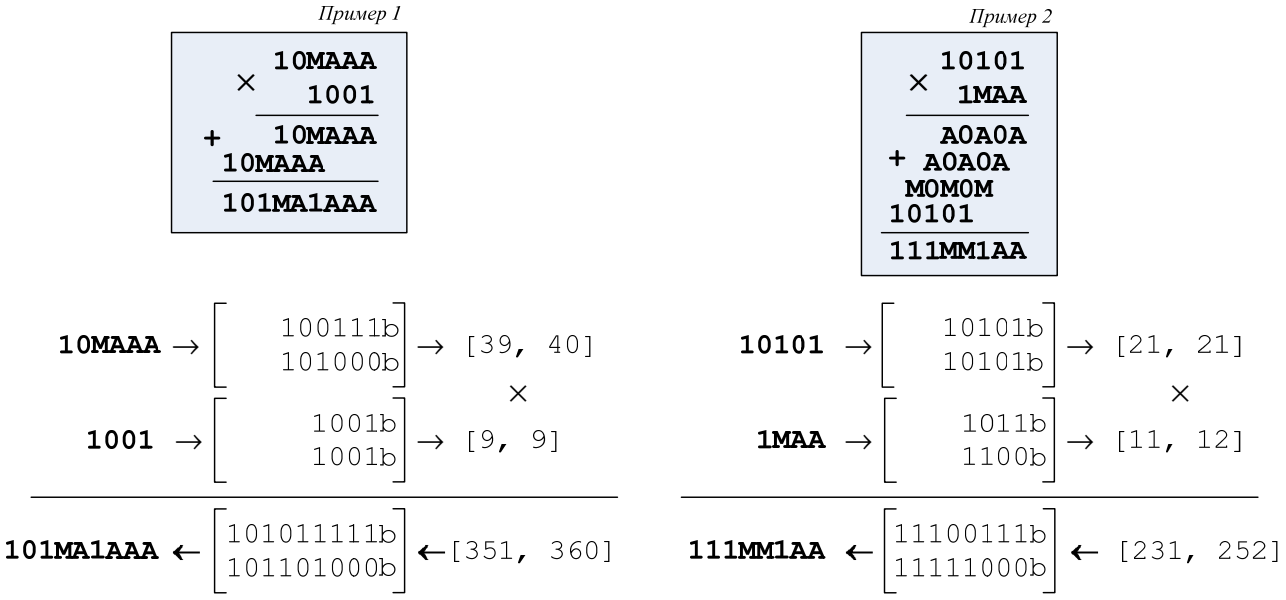


Рисунок Ж.3 – Примеры умножения тетракодов с демонстрацией проверки правильности полученного произведения

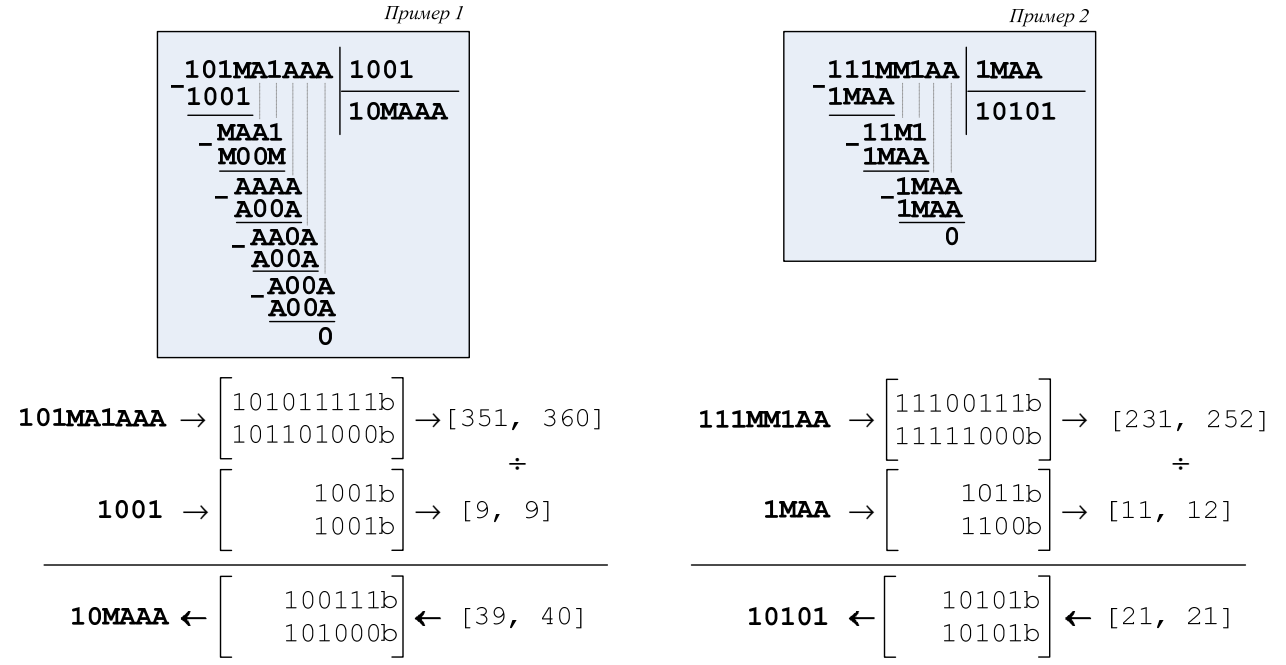


Рисунок Ж.4 – Примеры деления тетракодов с демонстрацией проверки правильности полученного частного

Приложение И

Модифицированные форматы чисел с плавающей запятой

Модифицированные форматы чисел с плавающей запятой (pbinary, или сокращенно pb) – форматы машинных чисел, аналогичные используемым в современных процессорах стандартным форматам (стандатт IEEE 754-2008). На рисунке ниже пред-ставлено соотношение модифицированных форматов (со стандарнтыми (binary) точности Ω с выделением полей знака, порядка, мантиссы, а также идентификатора формата.

По способу кодирования модифицированные форматы чисел делятся на бинарные и постбинарные форматы.

По типу кодируемого числа модифицируемые форматы чисел с плавающей запятой делятся на:

- *Числовые бинарные форматы* – это модифицированные форматы с плавающей запятой, с помощью которых кодируются вещественные числа и кодирование осуществляется средствами бинарного (двоичного) кодирования.

  По точности представления числовые бинарные форматы делятся на:

  - PostBinary 32;
  - PostBinary 64;
  - PostBinary 128;
  - PostBinary 256.

- *Дробные бинарные форматы* – это модифицированные форматы с плавающей запятой, с помощью которых кодируется пара вещественных чисел вида a/b, где a – числитель дроби, b – знаменатель дроби. Кодирование формата осуществляется средствами бинарного (двоичного) кодирования.

По точности представления дробные модифицированные бинарные форматы делятся на:

- PostBinary 32/16 f;
- PostBinary 64/32 f;
- PostBinary 128/64 f;
- PostBinary 256/128 f.

- *Интервальные бинарные форматы* – модифицированные форматы с плавающей запятой, с помощью которых кодируется пара вещественных чисел вида $[a_1, a_2]$, где a_1 – левая граница интервала, a_2 – правая граница интервала. Кодирование формата осуществляется средствами бинарного (двоичного) кодирования.

  По точности представления интервальные модифицированные бинарные форматы делятся на:

  - PostBinary 32/16 i;
  - PostBinary 64/32 i;
  - PostBinary 128/64 i;
  - PostBinary 256/128 i.

- *Числовые постбинарные форматы* – модифицированные форматы с плавающей запятой, с помощью которых кодируются вещественные числа и кодирование осуществляется средствами постбинарного (тетра-) кодирования.

  По точности представления числовые модифицированные постбинарные форматы делятся на:

  - PostBinary 32/16 p;
  - PostBinary 64/32 p;
  - PostBinary 128/64p;
  - PostBinary 256/128 p.

- *Дробные постбинарные форматы* – модифицированные форматы с плавающей запятой, с помощью которых кодируется пара вещественных чисел вида a/b, где a – числитель дроби, b – знаменатель дроби. Кодирование формата осуществляется средствами постбинарного (тетра-) кодирования.

  По точности представления дробные модифицированные постбинарные форматы делятся на:

  - PostBinary 64/16 fp;
  - PostBinary 128/32 fp;
  - PostBinary 256/64 fp.

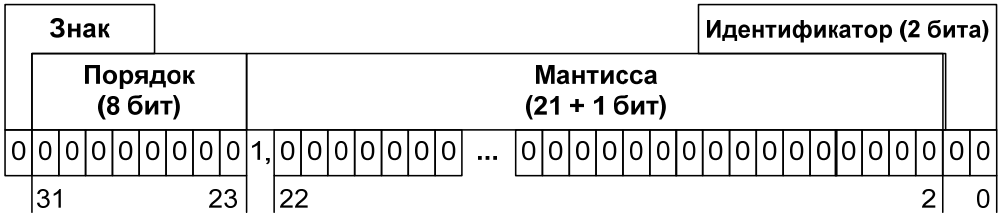
- *Интервальные постбинарные форматы* – модифицированные форматы с плавающей запятой, с помощью которых кодируется пара вещественных чисел $[a_1, a_2]$, где a_1 – левая граница интервала, a_2 – правая граница интервала. Кодирование формата осуществляется средствами постбинарного (тетра-) кодирования.

  По точности представления интервальные модифицированные постбинарные форматы делятся на:

  - PostBinary 64/16 ip;
  - PostBinary 128/32 ip;
  - PostBinary 256/64 ip.

Спецификация формата PostBinary 32:

Формат PostBinary 32 (pbinary32, pb32) предназначен для хранения одного вещественного числа (модификатор MF = 0h) одинарной точности (код формата CF = 0h) с использованем бинарного (двоичного) кодирования.



Размерность полей формата:	30 бит (1 бит + 8 бит + 21 бит)
знак	1 бит
мантисса	8 бит
порядок	21 бита
Поле идентификатора:	2 бита (MF = 1 бит + CF = 1 бит)
значение модификатора формата MF	0 – обозначает числовой формат и бинарное кодирование
значение кода формата CF	0 – обозначает поле данных длиной 32 бита
Смещение порядка:	+127

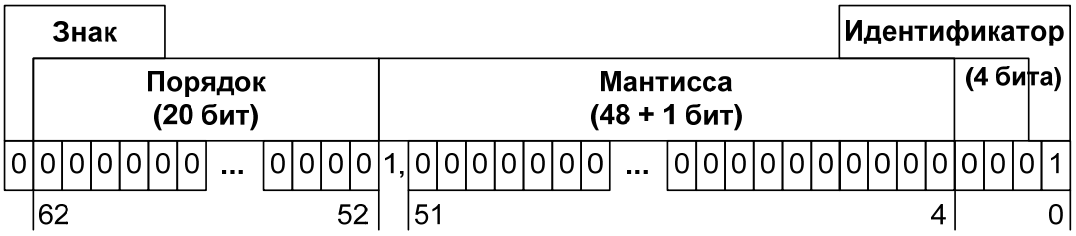
Исключительные числа:	**S**	**Exp**	**Mant**	**MF+CF**	**Значение**
положительный ноль	0	00000000	00000000000000000000000	00	+0.0
отрицательный ноль	1	00000000	00000000000000000000000	00	–0.0
положительная бесконечность	0	11111111	00000000000000000000000	00	+Infinity
отрицательная бесконечность	1	11111111	00000000000000000000000	00	–Infinity
не число	0	11111111	XXXXXXXXXXXXXXXXXXXXXXX	00	+NaN
не число	1	11111111	XXXXXXXXXXXXXXXXXXXXXXX	00	–NaN
	(где XXX..X – отличная от нуля мантисса)				

Числовые диапазоны:	**S**	**Exp**	**Mant**	**MF+CF**	**Значение**
мининальное денормализованное	X	00000000	00000000000000000000001	00	≈ ±5.6051939e–45
максимальное денормализованное	X	00000000	11111111111111111111111	00	≈ ±1.1754938e–38
мининальное нормализованное	X	00000001	00000000000000000000000	00	≈ ±1.1754944e–38
максимальное нормализованное	X	11111110	11111111111111111111111	00	≈ ±3.4028229e+38
	(X – значение разряда 0 или 1)				

Относительная точность десятичных цифр: 6

Спецификация формата PostBinary 64:

Формат PostBinary 64 (pb64) предназначен для хранения одного вещественного числа (модификатор MF = 0h) двойной точности (код формата CF = 1h) с использованием бинарного кодирования.



Размерность полей формата:	60 бит (1 бит + 11 бит + 48 бит)
знак	1 бит
мантисса	11бит
порядок	48 бит
Поле идентификатора:	4 бита (MF = 2 бита + CF = 2 бита)
значение модификатора формата MF	00 – обозначает числовой формат и бинарное кодирование
значение кода формата CF	01 – обозначает поле данных длиной 64 бита
Смещение порядка:	+1023

Исключительные числа:

	S	Exp	Mant	MF+CF
положительный ноль	0	00000000000	0000000000000...00	0001
отрицательный ноль	1	00000000000	0000000000000...00	0001
положительная бесконечность (+Infinity)	0	11111111111	0000000000000...00	0001
отрицательная бесконечность (–Infinity)	1	11111111111	0000000000000...00	0001
не число (+NaN)	0	11111111111	XXXXXXXXXXXXX...XX	0001
отрицательная бесконечность (–NaN)	1	11111111111	XXXXXXXXXXXXX...XX	0001

(где XXX..X – отличная от нуля мантисса)

Числовые диапазоны:

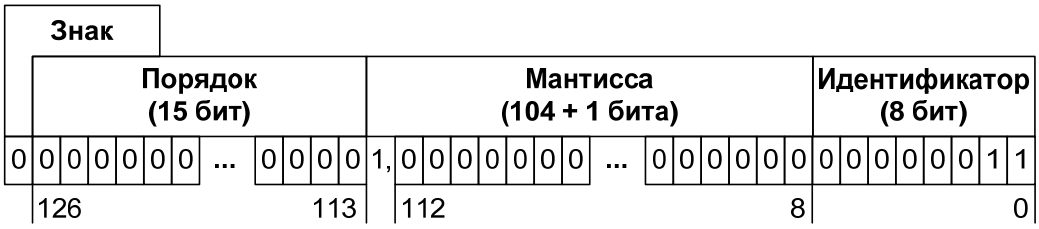
	S	Exp	Mant	MF+CF	Значение
мининальное денормализованное	X	00000000000	0000000000000...01	0001	≈ ±7.905050333499e–323
максимальное денормализованное	X	00000000000	1111111111111...11	0001	≈ ±2.2250738585072l9e–308
мининальное нормализованное	X	00000000001	0000000000000...00	0001	≈ ±2.225073858072e–308
максимальное нормализованное	X	11111111110	1111111111111...11	0001	≈ ±1.7976931348623e+308

(X – значение разряда 0 или 1)

Относительная точность десятичных цифр: 14–15

Спецификация формата PostBinary 128:

Формат PostBinary 128 (pb128) предназначен для хранения одного вещественного числа (модификатор MF = 00h) четверной точности (код формата CF = 3h) с использованием бинарного (двоичного) кодирования.



Размерность полей формата:	120 бит (1 бит + 15 бит + 104 бита)
знак	1 бит
мантисса	15 бит
порядок	104 бита
Поле идентификатора:	8 бит (MF = 5 бит + CF = 3 бита)
значение модификатора формата MF	00000 – обозначает числовой формат и бинарное кодирование
значение кода формата CF	011 – обозначает поле данных длиной 128 бит
Смещение порядка:	+16383

Исключительные числа:	**S**	**Exp**	**Mant**	**MF+CF**
положительный ноль	0	000000000000000	0000000000000...00	00000011
отрицательный ноль	1	000000000000000	0000000000000...00	00000011
положительная бесконечность (+Infinity)	0	111111111111111	0000000000000...00	00000011
отрицательная бесконечность (–Infinity)	1	111111111111111	0000000000000...00	00000011
не число (+NaN)	0	111111111111111	XXXXXXXXXXXXX...XX	00000011
отрицательная бесконечность (–NaN)	1	111111111111111	XXXXXXXXXXXXX...XX	00000011

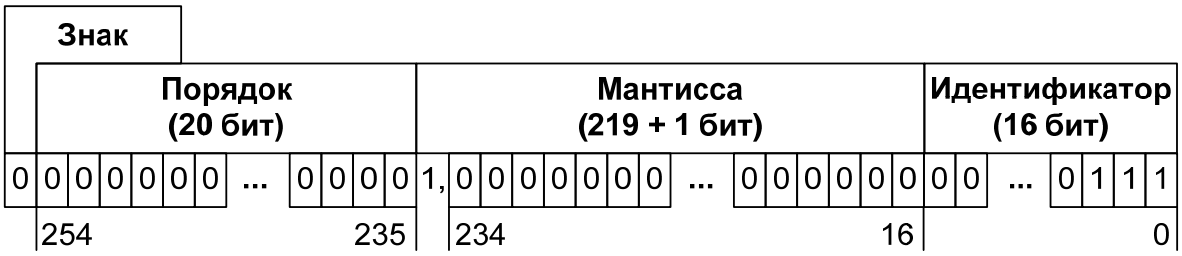
(где XXX..X – отличная от нуля мантисса)

Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ± 1.65764483057613442839665637330 63 e–4963
максимальное денормализованное	≈ ± 3.36210314311209350626267781732 16 e–4932
мининальное нормализованное	≈ ± 3.36210314311209350626267781732 18 e–4932
максимальное нормализованное	≈ ± 1.18973149535723176508575932662 80 e+4932

Относительная точность десятичных цифр: 31–32

Спецификация формата PostBinary 256:

Формат PostBinary 256 (pb256) предназначен для хранения одного вещественного числа (модификатор MF = 000h) восьмерной точности (код формата CF = 7h) с использованием бинарного (двоичного) кодирования.



Размерность полей формата:	240 бит (1 бит + 20 бит + 219 бит)
знак	1 бит
мантисса	20 бит
порядок	219 бит
Поле идентификатора:	16 бит (MF = 12 бит + CF = 4 бита)
значение модификатора формата MF	000000000000 – обозначает числовой формат и бинарное кодирование
значение кода формата CF	0111 – обозначает поле поле данных длиной 256 бит
Смещение порядка:	+524287

Исключительные числа:	**S**	**Exp**	**Mant**	**MF+CF**
положительный ноль	0	00000000000000000000	00000000000000...00	000000000000111
отрицательный ноль	1	00000000000000000000	00000000000000...00	000000000000111
положительная бесконечность (+Infinity)	0	11111111111111111111	00000000000000...00	000000000000111
отрицательная бесконечность (–Infinity)	1	11111111111111111111	00000000000000...00	000000000000111
не число (+NaN)	0	11111111111111111111	XXXXXXXXXXXXX...XX	000000000000111
отрицательная бесконечность (–NaN)	1	11111111111111111111	XXXXXXXXXXXXX...XX	000000000000111

(где XXX..X – отличная от нуля мантисса)

Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ± 1.828623360511354903950603959893667737399055001281 e–1578924
мининальное нормализованное	≈ ± 1.540612133552872047061497749015684650691656123143 e–157826
максимальное нормализованное	≈ ± 2.596370567831000776126596495726882827744734376348 e+157826

Относительная точность десятичных цифр: 66

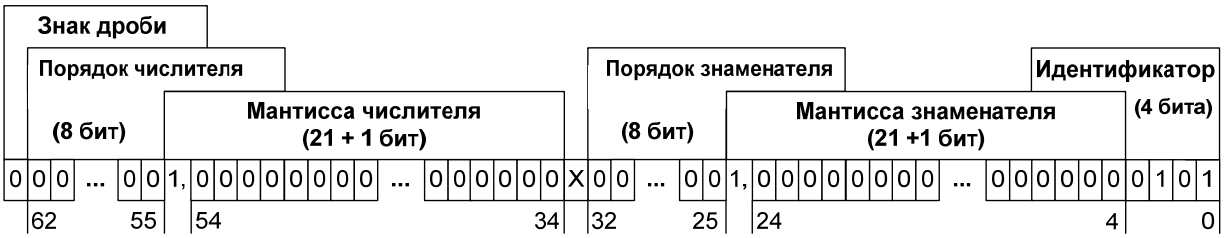
Спецификация формата PostBinary 32/16 f:

Формат PostBinary 32/16 f (pb32/16f) предназначен для хранения двух вещественных чисел (модификатор MF = 1h) половинной точности (код формата CF = 0h), представляющих собой числитель и знаменатель дроби. Используется бинарное (двоичное) кодирование.

Размерность полей формата:	30 бит: 2 × (1 бит + 5 бит + 9 бит)
знак	1 бит
мантисса	5 бит
порядок	9 бит
Поле идентификатора:	2 бита (MF = 1 бит + CF = 1 бит)
значение модификатора формата MF	1 – дробный (или интервальный) формат и бинарное кодирование
значение кода формата CF	0 – обозначает поле данных длиной 32 бита
Смещение порядка:	+15
Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ± 1.192 e–7
максимальное денормализованное	≈ ± 6.092 e–5
мининальное нормализованное	≈ ± 6.104 e–5
максимальное нормализованное	±65 472
Относительная точность десятичных цифр:	3

Спецификация формата PostBinary 64/32 f:

Формат PostBinary 64/32 f (pb64/32f) предназначен для хранения двух вещественных чисел (модификатор MF = 1h) одинарной точности (код формата CF = 1h), представляющих собой числитель и знаменатель дроби. Используется бинарное (двоичное) кодирование.



Размерность полей формата:	60 бит: 2 × (1 бит + 8 бит + 21 бит)
знак	1 бит
мантисса	8 бит
порядок	21 бит
Поле идентификатора:	4 бита (MF = 2 бита + CF = 2 бита)
значение модификатора формата MF	01 – дробный формат и бинарное кодирование
значение кода формата CF	01 – обозначает поле данных длиной 64 бита
Смещение порядка:	+127

Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ±5.6051939 e–45
максимальное денормализованное	≈ ±1.1754938 e–38
мининальное нормализованное	≈ ±1.1754944 e–38
максимальное нормализованное	≈ ±3.4028229 e+38

Относительная точность десятичных цифр: 6 (для числителя и знаменателя)

Спецификация формата PostBinary 128/64 f:

Формат PostBinary 128/64 f (pb128/64f) предназначен для хранения двух вещественных чисел (модификатор MF = 01h) двойной точности (код формата CF = 3h), представляющих собой числитель и знаменатель дроби. Используется бинарное (двоичное) кодирование.

Размерность полей формата:	120 бит: 2 × (1 бит + 11 бит + 48 бит)
знак	1 бит
мантисса	11 бит
порядок	48 бит
Поле идентификатора:	8 бит (MF = 5 бит + CF = 3 бита)
значение модификатора формата MF	00001 – дробный формат и бинарное кодирование
значение кода формата CF	011 – обозначает поле данных длиной 128 бит
Смещение порядка:	+1023
Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ±7.905050333499 e–323
максимальное денормализованное	≈ ±2.22507385850719 e–308
мининальное нормализованное	≈ ±2.225073858072 e–308
максимальное нормализованное	≈ ±1.7976931348623 e+308
Относительная точность десятичных цифр:	14–15 (для числителя и знаменателя)

Спецификация формата PostBinary 256/128 f:

Формат PostBinary 256/128 f (pb256/128f) предназначен для хранения двух вещественных чисел (модификатор MF = 001h) четверной точности (код формата CF = 7h), представляющих собой числитель и знаменатель дроби. Используется бинарное (двоичное) кодирование.

Размерность полей формата:	240 бит: 2 × (1 бит + 15 бит + 104 бит)
знак	1 бит
мантисса	15 бит
порядок	104 бит
Поле идентификатора:	16 бит (MF = 12 бит + CF = 4 бита)
значение модификатора формата MF	000000000001 – дробный формат и бинарное кодирование
значение кода формата CF	0111 – обозначает поле данных длиной 256 бит
Смещение порядка:	+16383
Числовые диапазоны:	**Значение**
минимальное денормализованное	≈ ± 1.65764483057613442839666563733063 e–4963
максимальное денормализованное	≈ ± 3.36210314311209350626267781732l6 e–4932
минимальное нормализованное	≈ ± 3.36210314311209350626267781732l8 e–4932
максимальное нормализованное	≈ ± 1.18973149535723176508575932662800 e+4932
Относительная точность десятичных цифр:	31–32 (для числителя и знаменателя)

Спецификация формата PostBinary 32/16 i:

Формат PostBinary 32/16 i (pb32/16i) предназначен для хранения двух вещественных чисел (модификатор MF = 1h) половинной точности (код формата CF = 0h), представляющих собой границы интервала. Используется бинарное (двоичное) кодирование.

Размерность полей формата:	30 бит: 2 × (1 бит + 5 бит + 9 бит)
знак	1 бит
мантисса	5 бит
порядок	9 бит
Поле идентификатора:	2 бита (MF = 1 бит + CF = 1 бит)
значение модификатора формата MF	1 – интервальный (или дробный) формат и бинарное кодирование
значение кода формата CF	0 – обозначает поле данных длиной 32 бита
Смещение порядка:	+15
Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ± 1.192 e–7
максимальное денормализованное	≈ ± 6.092 e–5
мининальное нормализованное	≈ ± 6.104 e–5
максимальное нормализованное	±65 472
Относительная точность десятичных цифр:	3 (для левой и правой границ)

Спецификация формата PostBinary 64/32 i:

Формат PostBinary 64/32 i (pb64/32i) предназначен для хранения двух вещественных чисел (модификатор MF = 2h) одинарной точности (код формата CF = 1h), представляющих собой границы интервала. Используется бинарное (двоичное) кодирование.

Размерность полей формата:	60 бит: 2 × (1 бит + 8 бит + 21 бит)
знак	1 бит
мантисса	8 бит
порядок	21 бит
Поле идентификатора:	4 бита (MF = 2 бита + CF = 2 бита)
значение модификатора формата MF	10 – интервальный формат и бинарное кодирование
значение кода формата CF	01 – обозначает поле данных длиной 64 бита
Смещение порядка:	+127

Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ±5.6051939 e–45
максимальное денормализованное	≈ ±1.1754938 e–38
мининальное нормализованное	≈ ±1.1754944 e–38
максимальное нормализованное	≈ ±3.4028229 e+38

Относительная точность десятичных цифр: 6 (для левой и правой границ)

Спецификация формата PostBinary 128/64 i:

Формат PostBinary 128/64 i (pb128/64i) предназначен для хранения двух вещественных чисел (модификатор MF = 02h) двойной точности (код формата CF = 3h), представляющих собой границы интервала. Используется бинарное (двоичное) кодирование.

Размерность полей формата:	120 бит: 2 × (1 бит + 11 бит + 48 бит)
знак	1 бит
мантисса	11 бит
порядок	48 бит
Поле идентификатора:	8 бит (MF = 5 бит + CF = 3 бита)
значение модификатора формата MF	00010 – интервальный формат и бинарное кодирование
значение кода формата CF	011 – обозначает поле данных длиной 128 бит
Смещение порядка:	+1023

Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ±7.905050333499 e–323
максимальное денормализованное	≈ ±2.22507385850719 e–308
мининальное нормализованное	≈ ±2.225073858072 e–308
максимальное нормализованное	≈ ±1.7976931348623 e+308

Относительная точность десятичных цифр: 14–15 (для левой и правой границ)

Спецификация формата PostBinary 256/128 i:

Формат PostBinary 256/128 i (pb256/128i) предназначен для хранения двух вещественных чисел (модификатор MF = 002h) четверной точности (код формата CF = 7h), представляющих собой границы интервала. Используется бинарное (двоичное) кодирование.

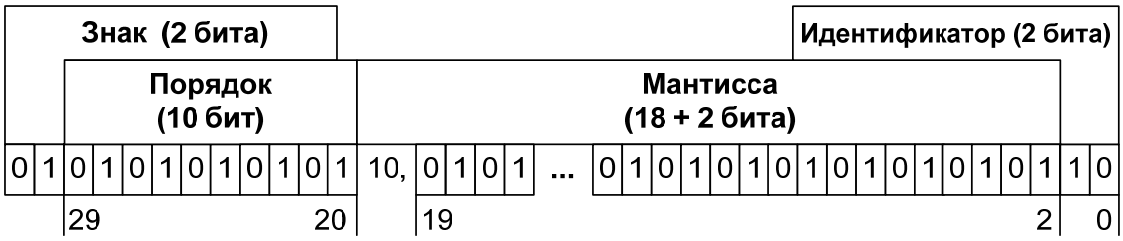
Размерность полей формата:	240 бит: 2 × (1 бит + 15 бит + 104 бит)
знак	1 бит
мантисса	15 бит
порядок	104 бит
Поле идентификатора:	16 бит (MF = 12 бит + CF = 4 бита)
значение модификатора формата MF	000000000010 – интервальный формат и бинарное кодирование
значение кода формата CF	0111 – обозначает поле данных длиной 256 бит
Смещение порядка:	+16383

Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ± 1.65764483057613442839665633733063 e–4963
максимальное денормализованное	≈ ± 3.36210314311209350626267781732l6 e–4932
мининальное нормализованное	≈ ± 3.362103143112093506262677817321 8 e–4932
максимальное нормализованное	≈ ± 1.189731495357231765085759326628 0 e+4932

Относительная точность десятичных цифр: 31–32 (для леовй и правой границ)

Спецификация формата PostBinary 32/16 p:

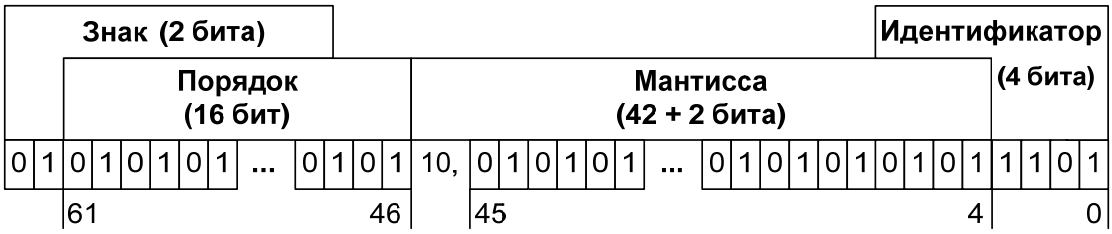
Формат PostBinary 32/16 p (pb32/16p) предназначен для хранения одного вещественного числа (модификатор MF = 1h) половинной точности (код формата CF = 0h) в виде тетракода, каждый разряд которого представлен двумя двоичными разрядами: 01 – тетраноль; 10 – тетраединица; 00 – неопределенное значение; 11 – множественное значение.



Размерность полей формата:	30 бит: 2 × (1 бит + 5 бит + 9 бит)
знак	2 бита
мантисса	10 бит
порядок	18 бит
Поле идентификатора:	2 бита (MF = 1 бит + CF = 1 бит)
значение модификатора формата MF	1 – числовой формат и постбинарное кодирование
значение кода формата CF	0 – обозначает поле данных длиной 32 бита
Смещение порядка:	+15
Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ± 1.192 e–7
максимальное денормализованное	≈ ± 6.092 e–5
мининальное нормализованное	≈ ± 6.104 e–5
максимальное нормализованное	±65 472
Относительная точность десятичных цифр:	3

Спецификация формата PostBinary 64/32 p:

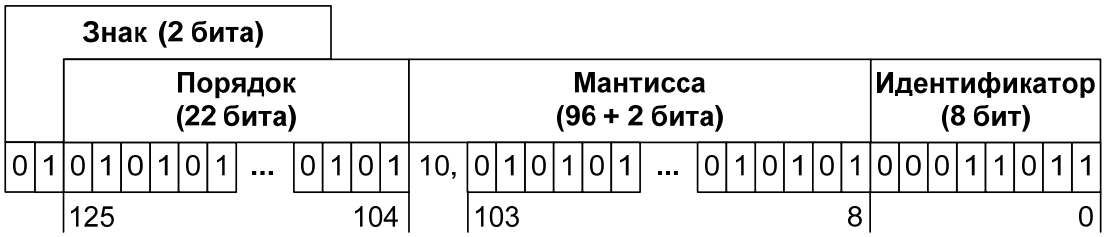
Формат PostBinary 64/32 p (pb64/32p) предназначен для хранения одного вещественного числа (модификатор MF = 3h) одинарной точности (код формата CF = 1h) в виде тетракода, каждый разряд которого представлен двумя двоичными разрядами: 01 – тетраноль; 10 – тетраединица; 00 – неопределенное значение; 11 – множественное значение.



Размерность полей формата:	60 бит: 2 × (1 бит + 8 бит + 21 бит)
знак	2 бита
мантисса	16 бит
порядок	42 бита
Поле идентификатора:	4 бита (MF = 2 бита + CF = 2 бита)
значение модификатора формата MF	11 – числовой формат и постбинарное кодирование
значение кода формата CF	01 – обозначает поле данных длиной 64 бита
Смещение порядка:	+127
Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ±5.6051939 e–45
максимальное денормализованное	≈ ±1.1754938 e–38
мининальное нормализованное	≈ ±1.1754944 e–38
максимальное нормализованное	≈ ±3.4028229 e+38
Относительная точность десятичных цифр:	6

Спецификация формата PostBinary 128/64 p:

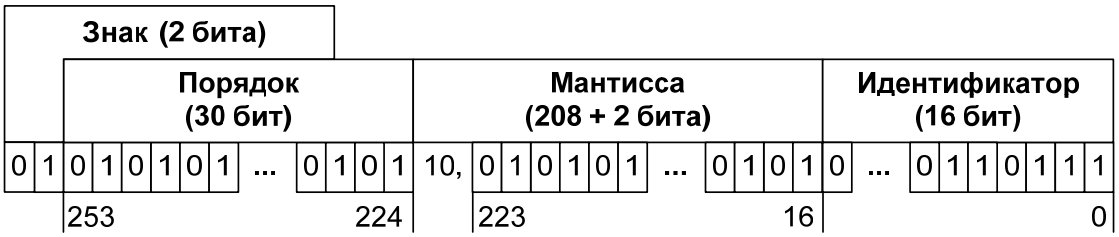
Формат PostBinary 128/64 p (pb128/64p) предназначен для хранения одного вещественного числа (модификатор MF = 03h) двойной точности (код формата CF = 3h) в виде тетракода, каждый разряд которого представлен двумя двоичными разрядами: 01 – тетраноль; 10 – тетраединица; 00 – неопределенное значение; 11 – множественное значение.



Размерность полей формата:	120 бит: 2 × (1 бит + 11 бит + 48 бит)
знак	2 бита
мантисса	22 бита
порядок	96 бит
Поле идентификатора:	8 бит (MF = 5 бит + CF = 3 бита)
значение модификатора формата MF	00011 – числовой формат и постбинарное кодирование
значение кода формата CF	011 – обозначает поле данных длиной 128 бит
Смещение порядка:	+1023
Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ±7.905050333499 e–323
максимальное денормализованное	≈ ±2.22507385850719 e–308
мининальное нормализованное	≈ ±2.225073858072 e–308
максимальное нормализованное	≈ ±1.7976931348623 e+308
Относительная точность десятичных цифр:	14–15

Спецификация формата PostBinary 256/128 p:

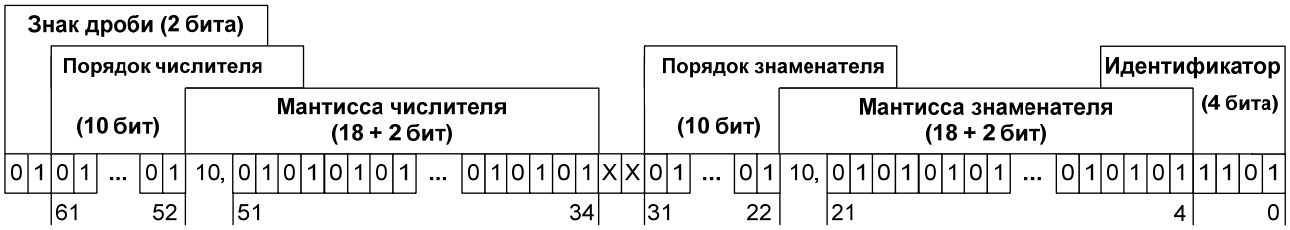
Формат PostBinary 256/128 p (pb256/128p) предназначен для хранения одного вещественного числа (модификатор MF = 003h) четверной точности (код формата CF = 7h) в виде тетракода, каждый разряд которого представлен двумя двоичными разрядами: 01 – тетраноль; 10 – тетраединица; 00 – неопределенное значение; 11 – множественное значение.



Размерность полей формата:	240 бит: 2 × (1 бит + 15 бит + 104 бит)
знак	2 бита
мантисса	30 бит
порядок	208 бит
Поле идентификатора:	16 бит (MF = 12 бит + CF = 4 бита)
значение модификатора формата MF	000000000011 – числовой формат и постбинарное кодирование
значение кода формата CF	0111 – обозначает поле данных длиной 256 бит
Смещение порядка:	+16383
Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ± 1.65764483057613442839666563733063 e–4963
максимальное денормализованное	≈ ± 3.36210314311209350626267781732l6 e–4932
мининальное нормализованное	≈ ± 3.36210314311209350626267781732l8 e–4932
максимальное нормализованное	≈ ± 1.18973149535723176508575932662800 e+4932
Относительная точность десятичных цифр:	31–32

Спецификация формата PostBinary 64/16 fp:

Формат PostBinary 256/128 fp (pb256/128fp) предназначен для хранения двух вещественных чисел (модификатор MF = 3h) половинной точности (код формата CF = 1h), представляющих собой числитель и знаменатель дроби в виде тетракода, каждый разряд которого представлен двумя двоичными разрядами: 01 – тетраноль; 10 – тетраединица; 00 – неопределенное значение; 11 – множественное значение.



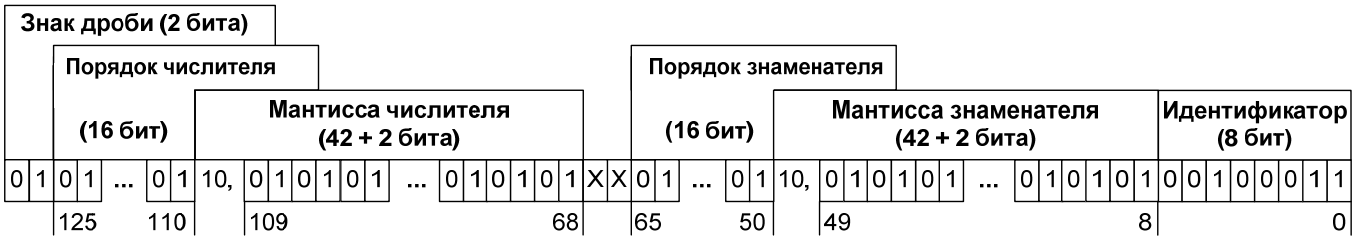
Размерность полей формата:	60 бит: 2 × (2 × (1 бит + 5 бит + 9 бит))
знак	2 бита
мантисса	10 бит
порядок	18 бит
Поле идентификатора:	4 бита (MF = 2 бита + CF = 2 бита)
значение модификатора формата MF	11 – дробный (или интервальный) формат и постбинарное кодирование
значение кода формата CF	01 – обозначает поле данных длиной 64 бита
Смещение порядка:	+15

Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ± 1.192 e–7
максимальное денормализованное	≈ ± 6.092 e–5
мининальное нормализованное	≈ ± 6.104 e–5
максимальное нормализованное	±65 472

Относительная точность десятичных цифр: 3 (для числителя и знаменателя дроби)

Спецификация формата PostBinary 128/32 fp:

Формат PostBinary 128/32 fp (pb128/32fp) предназначен для хранения двух вещественных чисел (модификатор MF = 04h) одинарной точности (код формата CF = 3h), представляющих собой числитель и знаменатель дроби в виде тетракода, каждый разряд которого представлен двумя двоичными разрядами: 01 – тетраноль; 10 – тетраединица; 00 – неопределенное значение; 11 – множественное значение.



Размерность полей формата:	120 бит: 2 × (2 × (1 бит + 8 бит + 21 бит))
знак	2 бита
мантисса	16 бит
порядок	42 бита
Поле идентификатора:	8 бит (MF = 5 бит + CF = 3 бита)
значение модификатора формата MF	0100 – дробный формат и постбинарное кодирование
значение кода формата CF	011 – обозначает поле данных длиной 128 бит
Смещение порядка:	+127
Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ±5.6051939 e–45
максимальное денормализованное	≈ ±1.1754938 e–38
мининальное нормализованное	≈ ±1.1754944 e–38
максимальное нормализованное	≈ ±3.4028229 e+38
Относительная точность десятичных цифр:	6 (для числителя и знаменателя дроби)

Спецификация формата PostBinary 256/64 fp:

Формат PostBinary 256/64 fp (pb256/64fp) предназначен для хранения двух вещественных чисел (модификатор MF = 004h) двойной точности (код формата CF = 7h), представляющих собой числитель и знаменатель дроби в виде тетракода, каждый разряд которого представлен двумя двоичными разрядами: 01 – тетраноль; 10 – тетраединица; 00 – неопределенное значение; 11 – множественное значение.

Размерность полей формата:	240 бит: 2 × (2 × (1 бит + 11 бит + 48 бит))
знак	2 бита
мантисса	22 бита
порядок	96 бит
Поле идентификатора:	16 бит (MF = 12 бит + CF = 4 бита)
значение модификатора формата MF	000000000100 – дробный формат и постбинарное кодирование
значение кода формата CF	0111 – обозначает поле данных длиной 256 бит
Смещение порядка:	+1023
Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ±7.905050333499 e–323
максимальное денормализованное	≈ ±2.22507385850719 e–308
мининальное нормализованное	≈ ±2.225073858072 e–308
максимальное нормализованное	≈ ±1.7976931348623 e+308
Относительная точность десятичных цифр:	14–15 (для числителя и знаменателя дроби)

Спецификация формата PostBinary 64/16 ip:

Формат PostBinary 64/16 ip (pb64/16ip) предназначен для хранения двух вещественных чисел (модификатор MF = 3h) половинной точности (код формата CF = 1h), представляющих собой границы интервала в виде тетракода, каждый разряд которого представлен двумя двоичными разрядами: 01 – тетраноль; 10 – тетраединица; 00 – неопределенное значение; 11 – множественное значение.

Размерность полей формата:	60 бит: 2 × (2 × (1 бит + 5 бит + 9 бит))
знак	2 бита
мантисса	10 бит
порядок	18 бит
Поле идентификатора:	4 бита (MF = 2 бита + CF = 2 бита)
значение модификатора формата MF	11 – интервальный (или дробный формат и постбинарное кодирование
значение кода формата CF	01 – обозначает поле данных длиной 64 бита
Смещение порядка:	+15
Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ± 1.192 e–7
максимальное денормализованное	≈ ± 6.092 e–5
мининальное нормализованное	≈ ± 6.104 e–5
максимальное нормализованное	± 65 472
Относительная точность десятичных цифр:	3 (для каждой границы интервала)

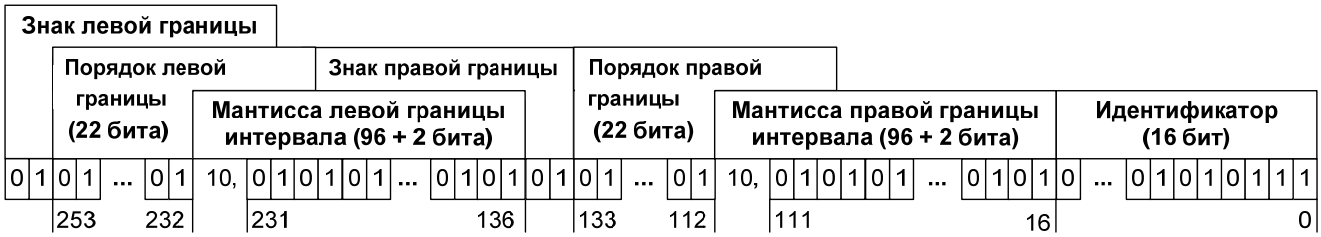
Спецификация формата PostBinary 128/32 ip:

Формат PostBinary 128/32 ip (pb128/32ip) предназначен для хранения двух вещественных чисел (модификатор MF = 05h) одинарной точности (код формата CF = 3h), представляющих собой границы интервала в виде тетракода, каждый разряд которого представлен двумя двоичными разрядами: 01 – тетраноль; 10 – тетраединица; 00 – неопределенное значение; 11 – множественное значение.

Размерность полей формата:	120 бит: 2 × (2 × (1 бит + 8 бит + 21 бит))
знак	2 бита
мантисса	16 бит
порядок	42 бита
Поле идентификатора:	8 бит (MF = 5 бит + CF = 3 бита)
значение модификатора формата MF	0101 – интервальный формат и постбинарное кодирование
значение кода формата CF	011 – обозначает поле данных длиной 128 бит
Смещение порядка:	+127
Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ±5.6051939 e–45
максимальное денормализованное	≈ ±1.1754938 e–38
мининальное нормализованное	≈ ±1.1754944 e–38
максимальное нормализованное	≈ ±3.4028229 e+38
Относительная точность десятичных цифр:	6 (для каждой границы интервала)

Спецификация формата PostBinary 256/64 ip:

Формат PostBinary 256/64 ip (pb256/64ip) предназначен для хранения двух вещественных чисел (модификатор MF = 005h) двойной точности (код формата CF = 7h), представляющих собой границы интервала в виде тетракода, каждый разряд которого представлен двумя двоичными разрядами: 01 – тетраноль; 10 – тетраединица; 00 – неопределенное значение; 11 – множественное значение.

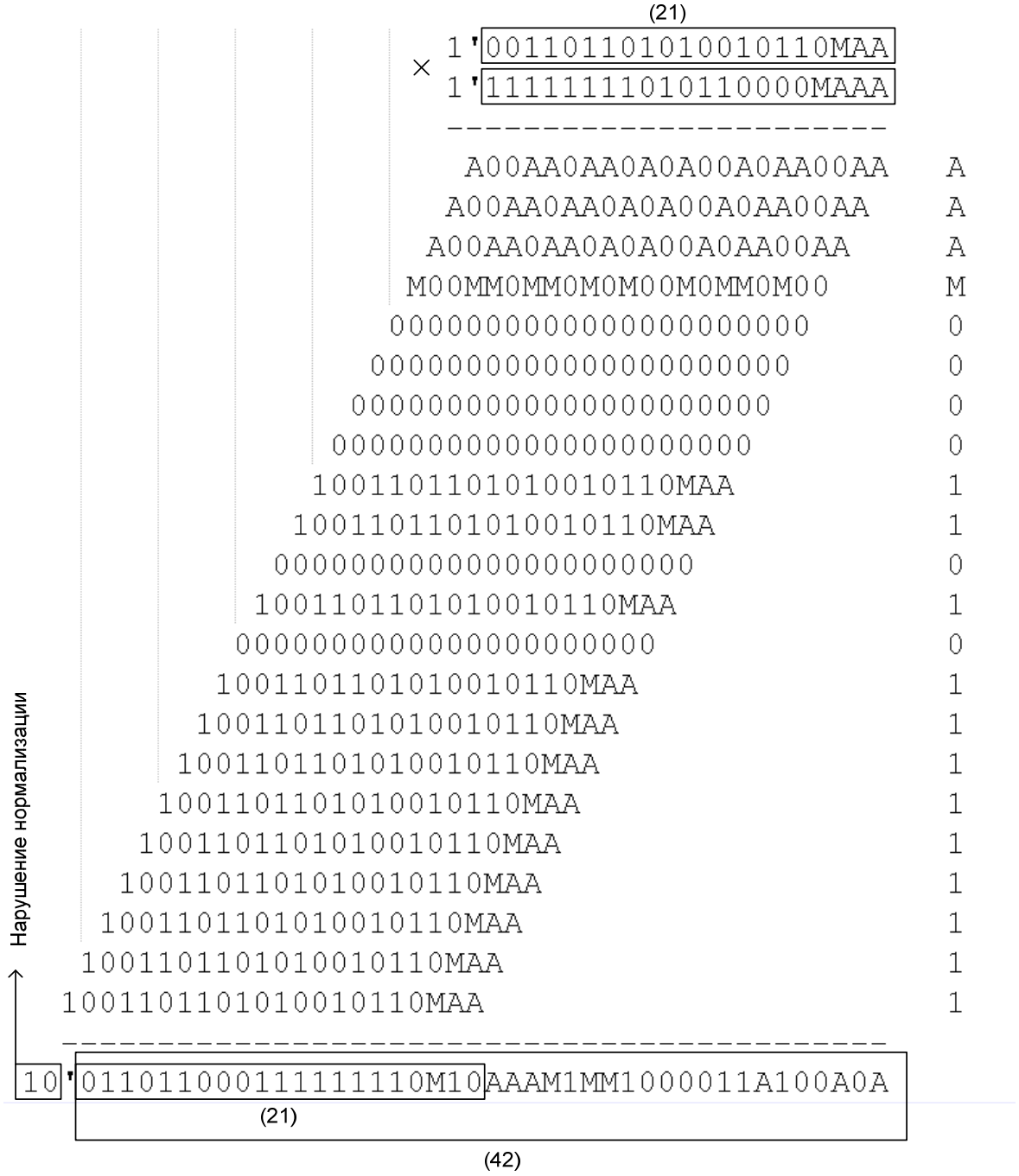


Размерность полей формата:	240 бит: 2 × (2 × (1 бит + 11 бит + 48 бит))
знак	2 бита
мантисса	22 бита
порядок	96 бит
Поле идентификатора:	16 бит (MF = 12 бит + CF = 4 бита)
значение модификатора формата MF	000000000101 – интервальный формат и постбинарное кодирование
значение кода формата CF	0111 – обозначает поле данных длиной 256 бит
Смещение порядка:	+1023
Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ±7.905050333499 e–323
максимальное денормализованное	≈ ±2.22507385850719 e–308
мининальное нормализованное	≈ ±2.225073858072 e–308
максимальное нормализованное	≈ ±1.7976931348623 e+308
Относительная точность десятичных цифр:	14–15 (для каждой границы интервала)

Приложение К

Пример умножения мантисс чисел формата pb128/32ip

(в столбце справа размещены для наглядности текущие разряды множителя, на который в каждом шаге умножается множимое)



Приложение Л

Копии документов о внедрении результатов исследований

ДОНЕЦКАЯ НАРОДНАЯ РЕСПУБЛИКА
МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
"ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ"
283001, г. Донецк, ул. Артема, 58 тел.: (062) 337-17-33, 335-75-62, факс: (062) 304-12-78
эл. почта: donntu.info@mail.ru

16.10.19 № 01-508/27

На № _____

СПРАВКА

о внедрении результатов исследований диссертационной работы Иваницы Сергея Васильевича «Обоснование закономерностей, арифметико-логических алгоритмов и структур систем компьютерной обработки информации», представленной на соискание ученой степени кандидата технических наук по специальности 05.13.01 – «Системный анализ, управление и обработка информации» (технические науки)

В ГОУВПО «Донецкий национальный технический университет» приняты к внедрению в учебный процесс и используются при чтении лекций и проведении практических лабораторных занятий для подготовки бакалавров на кафедре «Компьютерная инженерия» (КИ) по дисциплине «Арифметико-логические основы цифровых автоматов» (н. п. 09.03.01. «Информатика и вычислительная техника» для специальностей «Вычислительные машины, комплексы, системы и сети» и «Программное обеспечение средств вычислительной техники») следующие разработки, полученные в диссертационной работе Иваницы С. В.:

– основные закономерности тетралокики, построение функционально полных базисов на основе унарных и бинарных тетрафункций;

– операционные схемы и алгоритмы арифметических операций с тетракодами;

– проектирование аппаратных компонентов, реализующих функции тетралогики, построение на базе суммирующих компонентов, и, в совокупности, – создание операционных узлов для работы с постбинарными данными.

Исследования, полученные в диссертационной работе Иваницы С. В., также включены в учебное пособие по дисциплине «Арифметико-логические основы цифровых автоматов»: Иваница, С. В. Арифметические основы вычислительных систем. Арифметика чисел с фиксированной запятой : учеб. пособие для вузов / С. В. Иваница ; ГОУВПО "ДОННТУ". — Донецк : Технопарк ДОННТУ «УНИТЕХ», 2018. — 360 с.

Первый проректор А. А. Каракозов

Нач. учебного отдела Б. В. Гавриленко

Зав. кафедрой КИ А. Я. Аноприенко

М. П.

Соответствует оригиналу
Ученый секретарь Д 01.024.04
Т.В. Завадская

Государственное предприятие «СТИРОЛ»

Донецкая Народная Республика, 84610, г. Горловка, Калининский район,
ул. Горловской дивизии, 10, тел.: (06242) 4-41-18, эл. почта: trade@stirol.su.
Идентификационный код юридического лица: 51019276,
код банка: 400019, р/с: 26005721050100 в ЦРБ ДНР

№ 327/4 от 18.10.19г.

СПРАВКА

о внедрении результатов исследований диссертационной работы Иваницы Сергея Васильевича «Обоснование закономерностей, арифметико-логических алгоритмов и структур систем компьютерной обработки информации», представленной на соискание ученой степени кандидата технических наук по специальности 05.13.01 – «Системный анализ, управление и обработка информации» (технические науки)

На государственном предприятии «Стирол» (ГП «Стирол», г. Горловка, ДНР) с целью дальнейшего развития работ по совершенствованию технологических процессов и параметров оборудования, осуществляющего процессы для производства изделий из полимеров, приняты к использованию следующие разработки и рекомендации, полученные в диссертационной работе Иваницы С. В.:

– алгоритмы и программы расчета достоверных конструкционных параметров оборудования, с помощью которого обеспечивается нужная деформация сжатия, необходимая для фиксации вытянутой структуры полиэтилентерефталата, полипропиленовой и полиэтиленовой пленок;

– использование модифицированной арифметики для построения математической модели процесса регулирования вязкости полимерных материалов в зависимости от градиента скорости сдвига при различных температурах;

– использование специализированных типов данных для повышения эффективности компьютерных расчетов расходных коэффициентов в производстве пленок на основе ПВХ методом каландрования;

– рекомендации по модификации параметров, используемых при формировании расчетных характеристик для целенаправленного изменения физико-механических и химических свойств полимеров, сочетания в одном материале противоположных качеств, например твердости и гибкости, что повышает качество выпускаемого конечного продукта.

Директор
ГП «Стирол» М. В. Чепак

М. П.

Соответствует оригиналу
Ученый секретарь Д 01.024.04
Т.В. Завадская