Государственное образовательное учреждение
высшего профессионального образования
«Донецкий национальный
технический университет»

А. Я. Аноприенко,

С. В. Иваница

ВВЕДЕНИЕ В ПОСТБИНАРНЫЙ КОМПЬЮТИНГ

АРИФМЕТИКО-ЛОГИЧЕСКИЕ ОСНОВЫ И ПРОГРАММНО-АППАРАТНАЯ РЕАЛИЗАЦИЯ

М о н о г р а ф и я

Донецк, 2017

УДК 004.2
ББК 32.97
А69

Рецензенты:

Павлыш В. Н., д. т. н., профессор, заведующий кафедрой прикладной математики ДонНТУ;

Толстых В. К., д. т. н., профессор кафедры компьютерных технологий Донецкого национального университета;

Аверин Г. В., д. т. н., заведующий кафедрой общей математики Белгородского государственного национального исследовательского университета.

Публикуется в соответствии с решением Ученого совета Донецкого национального технического университета № 5 от 2 июня 2017 года.

Аноприенко А. Я., **Иваница С. В.**

А69 Введение в постбинарный компьютинг. Арифметико-логические основы и программно-аппаратная реализация / А. Я. Аноприенко, С. В. Иваница — Донецк: ДонНТУ, УНИТЕХ, 2017. — 308 с.

ISBN 978–966–8248–75–7

Монография посвящена рассмотрению основных вопросов кодо-логической эволюции и перехода к постбинарному компьютингу на основе постбинарной логики и компьютерной арифметики (постбинарные вычисления, тетроарифметика). В книге также рассматривается аппаратная и программная реализация элементов и узлов постбинар-ных компьютерных систем на базе расширенного кодо-логического базиса. Материалы монографии предназначены для исследователей, специалистов, аспирантов и магистрантов, специализирующихся в области компьютерных наук и технологий.

УДК 004.2
ББК 32.97
А69

ISBN 978–966–8248–75–7

СОДЕРЖАНИЕ

Введение

Постбинарный компьютинг — 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.

В настоящее время компьютерные технологии будущего ассоциируются в первую очередь с квантовыми компьютерами и другими весьма экзотическими на сегодня решениями. Несомненно, квантовые вычисления рано или поздно станут обыденной реальностью и позволят в целом ряде областей применения кардинально решить проблему повышения производительности обработки информации. Но, во-первых, ввиду необходимости решения множества физических и технических проблем достижение данной цели отнюдь не гарантировано и возможно лишь после многих десятилетий исследований и разработок. Во-вторых, квантовые вычисления это в действительности лишь одно из множества направлений современного технологического развития, также не гарантирующих возможность нахождения универсальных решений, способных в корне изменить облик всех компьютерных технологий будущего. Но, как показали проведенные авторами исследования, некоторые важные особенности квантовых вычислений, связанные с кодированием

информации и возможностями распараллеливания ее обработки, могут быть реализованы уже на базе существующих технологий путем реализации идей и методов постбинарного компьютинга. Именно поэтому постбинарный компьютинг на начальных этапах его развития предлагалось назвать **квазиквантовым компьютингом**. Отказаться от такого названия пришлось лишь потому, что, во-первых, к физической сути квантовых вычислений в том виде, как это представляется сегодня, постбинарный компьютинг в действительности прямого отношения не имеет, а, во-вторых, более важным представляется отразить в названии тот факт, что мы имеем дело с новым закономерным этапом кодо-логической эволюции информационных технологий, а не просто с какими-либо изобретениями или идеями, характер появления которых может быть относительно субъективным и/или случайным.

Одной из важнейших будущих альтернатив квантовым компьютерам являются так называемые ДНК-компьютеры, которые предполагается реализовать на основе новейших достижений нанобиоэлектроники. Сутью этих гипотетических компьютеров будущего является использование в качестве средства хранения и обработки информации молекул ДНК, обеспечивающих передачу из поколения в поколение и реализацию генетической программы развития и функционирования живых организмов. Считается, что преодоление технологического порога в 10 нм, с одной стороны, существенно приблизило исчерпание возможностей кремниевых технологий, но, с другой стороны, позволило перевести в практическую плоскость создание биологических компьютеров. И здесь не возникает сомнений, что за считанные десятилетия основные задачи создания биологических или ДНК-

компьютеров будут решены. Но, как и в случае с квантовыми компьютерами, биокомпьютеры, скорее всего, лишь займут свою определенную нишу и вряд ли сколь-нибудь существенно будут определять облик всего компьютинга будущего. Однако, многие идеи и решения, которые сформируются в процессе создания биокомпьютеров несомненно повлияют и на развитие информационно-компьютерных технологий в целом. В этой связи важно отметить, что изначально идеи постбинарного компьютинга зародились в ходе поиска средств и методов кодирования цифровой информации, аналогичных по своей эффективности генетическому кодированию. Именно поэтому самым первым вариантом названия для постбинарного компьютинга было его определение как **квазигенетического**. В дальнейшем от этого названия как основного также пришлось отказаться в пользу более общего и существенно более точного определения, особенно в контексте концепции кодо-логической эволюции. Но, тем не менее, с учетом всего изложенного выше можно считать, что в определенной степени **постбинарный компьютинг — это квазиквантовый и квазигенетический компьютинг.**

На первый вгляд нововведения, предлагаемые в контексте перехода к постбинарному компьютингу, не выглядят ни особо важными, ни революционными: расширение логического пространства, использование новых логических и цифровых значений – все это 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 частности, речь идет о следующих новых возможностях, достигаемых, как правило, всего лишь двукратным увеличением размеров кода и использованием алгоритмов постбинарной обработки:

Во-первых, внедрение в компьютерное представление числа информации о его реальной текущей точности. 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 самым различным путем (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. Различные частные решения такого рода проблем на сегодня имеются. Можно, например, упомянуть о так называемых лингвистических переменных, введенных Л. Заде и достаточно адекватно отражающих реальный уровень точности задаваемых экспертным путем значений с возможностью учета этой точности в последующих вычислениях. Но используются

такие технологии на сегодня весьма редко. Постбинарный компьютинг позволяет ввести тотальный контроль текущей точности всех количественных значений, что представляется абсолютно необходимым в контексте наблюдаемых в настоящее время тенденций цифровизации.

Во-вторых, 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**, что позволит существенно более компактно и адекватно кодировать информацию об объектах и процессах реального мира.

В-третьих, обеспечение естественного параллелизма обработки информации при работе с кодами, содержащими значения множественности, что значительно повышает эффективность разработки и реализации соответствующих алгоритмов, в полной мере использующих тотальный параллелизм современных и, особенно, будущих компьютерных систем.

В-четвертых, наряду 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.

В-пятых, не отменяя существующие технологии прабинарного и бинарного компьютинга, а значительно их расширяя и дополняя, **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.

Первая монография авторов по постбинарному компьютингу под названием «Постбинарный компьютинг и интервальные вычисления в контексте кодо-логической эволюции» вышла в 2011 году. В последующий период удалось существенно продвинуться в разработке арифметико-логических основ постбинарного компьютинга, а также — программных, программно-аппаратных и аппаратных решений, практически реализующих постбинарные вычисления. Проведенные исследования позволили более отчетливо увидеть и понять роль и значение постбинарного этапа в развитии компьютерных технологий. Все это и стало основой новой монографии, задуманной как краткое введение в постбинарный компьютинг, позволяющее понять основные закономерности его появления и особенности реализации.

В первой главе детально рассматривается формирование современных представлений о постбинарном компьютинге в контексте общей истории информационных и вычислительных технологий, а также в контексте развития идей кодо-логической эволюции и многомерного логического пространства.

Во второй главе рассматриваются арифметико-логические основы постбинарного компьютинга на примере основного варианта его реализации в ближайшем будущем, основанного на тетралогике и тетракодах.

В третьей главе представлены примеры программной и аппаратной реализации постбинарного кодирования и постбинарных вычислений, демонстрирующих возможности и особенности реализации идей постбинарного компьютинга на базе существующих компьютерных технологий.

Глава 1

ФОРМИРОВАНИЕ СОВРЕМЕННЫХ ПРЕДСТАВЛЕНИЙ О ПОСТБИНАРНОМ КОМПЬЮТИНГЕ

Прошлое и настоящее — наши средства,
только будущее — наша цель
Блез Паскаль

1.1. Зарождение идеи постбинарного компьютинга

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 авторы сформировали свое видение будущего компьютинга, объясняя и обосновывая его как неизбежное проявление долговременных тенденций и закономерностей развития информационных и компьютерных технологий.

Концепция постбинарного компьютинга в том виде, как она излагается в данном монографии, зародилась на основе синтеза и развития идей расширенного кодо-логического базиса, кодо-логической эволюции и квазигенетического кодо-логического компьютинга. Эти идеи в свою очередь сформировались в период защиты диссертации по высокопроизводительным системам компьютерной графики в 1987 году в послечернобыльском Киеве в Институте проблем моделирования в энергетике (ИПМЭ) Академии наук Украины одним из авторов [1]. Под влиянием чернобыльской катастрофы в ИПМЭ интенсифицировалось проведение научных семинаров, где не только всесторонне анализировались причины произошедшего в Чернобыле, но всесторонне рассматривались различные последствия катастрофы.

В числе прочих высказывались и предположения о том, что радиоактивное заражение, ставшее следствием Чернобыльской катастрофы, помимо всех негативных последствий может также активизировать и определенные эволюционные процессы и изменения. Открытым оставался вопрос о сути и значимости этих изменений, а также — о возможности их компьютерного моделирования. Все это породило повышенный интерес к вопросам эволюции. Как нельзя кстати пришлась только что изданная книга академика Н.Н. Моисеева «Алгоритмы развития» [2]. Автор книги к тому времени был известен как один из ведущих специалистов в области системного анализа и моделирования последствий ядерной войны [3, 4].

В работе «Алгоритмы развития» на основе идей В. И. Вернадского детально анализировался мировой эволюционный процесс, на основании чего делался вывод об общности процессов, протекающих в неживой материи,

в биоте и обществе. При этом также обосновывалось существенное значение информатики и вычислительной техники в реализации принципа коэволюции биосферы и общества.

Акцентировалось также внимание на том, что за последние десятилетия было сделано несколько «эпохальных открытий, позволяющих уточнить учение В. И. Вернадского и связать воедино многие факты, которые до этого носили фрагментарный характер. Во-первых, совсем недавно были обнаружены следы жизни на Земле, которая существовала 3,5–3,8 млрд. лет тому назад. Другими словами, возникновение Земли как космического тела и появление на ней жизни произошли, по космическим масштабам, почти одновременно. Этот факт переоценить невозможно! Другое открытие не меньшей значимости — это доказательство существования на Земле генетического кода, единого для всего живого. Единый алфавит из четырех букв…» (рис. 1.1).

Естественно возник вопрос о возможности формирования такого же эффективного «алфавита из четырех букв» для использования в современных информационно-компьютерных технологиях, которые в настоящее время основаны фактически всего лишь на 2-х «буквах»: 0 и 1. К использованию только этих двух «букв» или значений может быть однозначно сведено все современное разнообразие компьютерной логики, арифметики и систем счисления.

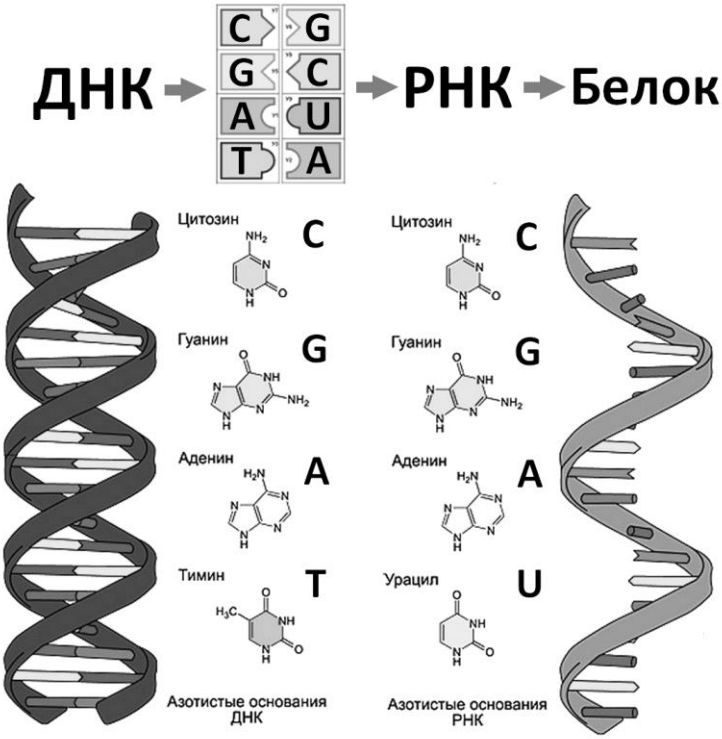


Рисунок 1.1 — Согласно модели Уотсона и Крика, сформированной к началу 1960-х годов при участии физика Георгия Гамова и других исследователей [5], молекула ДНК (дезоксирибонуклеиновой кислоты) состоит из двух полимерных цепочек. Каждая цепочка построена из звеньев четырех типов: C (цитозиновое), G (гуаниновое), A (адениновое) и T (тиминовое). Этот четырехзначный алфавит позволяет кодировать все ошеломляющее разнообразие жизни и постоянно ее воспроизводить путем синтеза белка на базе РНК (рибонуклеиновой кислоты), алфавит которой также четырехбуквенный (но вместо тимина урацил).

Размышления о том, что могло бы стать теми самыми дополнительными 2-мя «буквами» будущих компьютерных систем, которые позволили бы приблизить эффективность компьютерного кодирования информации к генетическому, довольно быстро привели к выводу, что этими новыми значениями должны стать: во-первых, «сверхзначение», представляющее 0 и 1 одновременно, и, во-вторых, «вероятностное значение», означающее, что при каждом конкретном использовании такого кода (аналогично процессу репликации ДНК) случайным образом будет формироваться значение 0 или 1.

Такие нововведения позволяли уже на уровне элементарного кодирования получать два принципиально новых качества: множественность кодируемых значений и возможность их неопределенности или изменчивости в некоторых заданных пределах.

Простейшей аналогией, поясняющей необходимость использования таких дополнительных значений для приближения компьютерного кодирования к сложности реального мира, является ситуация с бросанием монеты. В упрощенном варианте, полностью соответствующем современной двоичной логике и арифметике, предполагается, что результатом экспериментов с бросанием монеты может быть только «орел» и «решка», т.е. монета после броска может лечь только плашмя, а мы при этом будем однозначно видеть то, какой именно стороной она лежит вверх. Но в реальной жизни, хотя и очень редко, могут случиться и два других варианта: или монета после падения станет в вертикальное положение, когда будут видны одновременно обе ее стороны, или может навсегда закатиться в какую-нибудь щель, где мы ее уже никогда не увидим, а значит альтернатива «орел или решка» обернется полной неопределенностью.

В процессе последующего поиска аналогичных идей по развитию кодирования компьютерной информации обнаружилось, что примерно такое же решение предлагал в 1976 году Н. Белнап в работе «Как нужно рассуждать компьютеру», опубликованной на русском языке в 1981 году [6]. Свою публикацию он начал с очень осторожного утверждения: «Я предполагаю, что иногда должна использоваться четырехзначная логика». И далее он ведет изложение в столь же осторожном стиле: «Существуют обстоятельства, при которых некто — не вы — должен отказаться от обычной двузначной логики и использовать вместо нее другую». В конечном итоге он приходит к выводу, что более совершенная компьютерная логика должна использовать не два традиционных значения, а четыре: T – «некто говорит только Истину»; F – «некто говорит только Ложь», None — «не говорит ни Истины, ни Лжи»; Both — «говорит и Истину, и Ложь».

Сколь-нибудь значимого резонанса работа Н. Белнапа не получила и степень ее влияния на компьютерные технологии того времени оказалась практически нулевой. Но стало ясно, что идея уже «носится в воздухе» и начали созревать условия для ее практической реализации. В последнее время, например, распространение аналогичных подходов можно обнаружить в философской логике. В частности, доктор философских наук Бахтияров К.И. одну из своих публикаций завершает весьма оптимистичным предположением, что «ГЕНЕТИЧЕСКИЙ ПОДХОД В ЛОГИКЕ позволит воочию увидеть замысел Природы» [7].

1.2. Развитие идей постбинарного компьютинга

Значимой вехой стала стажировка одного из авторов в Штутгартском университете (Германия) в 1989–90 гг. при

поддержке Немецкой службы академических обменов (DAAD) и под руководством профессора Андреаса Ройтера, основателя и первого директора Института высокопроизводительных и распределенных компьютерных систем, ставшего основой одного из крупнейших в Европе центра суперкомпьютерных вычислений. В этот период первичные идеи постбинарного компьютинга оформились в своеобразную персональную исследовательскую программу, суть которой была впервые доложена и обсуждена на одном из семинаров под руководством А. Ройтера.

В дальнейшем в рамках этой программы сформировались концепции универсальных моделирующих сред и **расширенного эволюционирующего кодо-логического базиса**, суть которого сводилась к тому, что в информационно-вычислительных технологиях кодирование информации, логика и арифметика в действительности существенно взаимозависимы и развиваются в тесной и практически неразрывной связи друг с другом [8].

В процессе последующих исследований и разработок всё это постепенно трансформировалось в концепцию **квазигенетического постбинарного компьютинга** [9], основой которого являются тетралогика и тетракоды, позволяющие, в частности, реализовать вычисления с контролируемой и гибкой разрядностью. При этом в самом широком смысле в понятие «постбинарный компьютинг» входит в действительности все, что преодолевает ограничения двузначной логики и двоичной арифметики, в том числе троичная логика и троичный компьютинг, получившие определенной признание и распространение уже в XX веке. Но особая значимость постбинарного компьютинга заключается в том, что одна из его

разновидностей, основанная на четырехзначных (тетра) логике и арифметике, представляет особый интерес с точки зрения не столько современных, сколько будущих информационно-компьютерных технологий, так как позволяет вывести информационно-вычислительные технологии на качественно новый уровень. А так как на этом уровне наблюдается существенное приближение к свойствам и особенностям генетического кода, то средства кодирования информации и вычислений на основе тетралогики и тетракодов были названы квазигенетическими.

В 90-е годы начали публиковаться первые результаты исследований в области постбинарного компьютинга [10–13]. Существенными стимулом дальнейших исследований в данном направлении (особенно с учетом острейшего кризиса 90-х во всей постсоветской науке) стали и некоторые последующие стажировки в Штутгартском университете. В частности, основные идеи квазигенетической логики и квазигенетического кодирования окончательно сформировались в период научной стажировки одного из авторов в Институте высокопроизводительных и распределенных компьютерных систем Штутгартского университета в 1994–95 гг. Особую роль при этом сыграло участие в научном семинаре 28 октября 1994 года, посвященном 5-летнему юбилею Института.

Представленные на этом семинаре яркие доклады профессоров А. Ройтера (директора Института) и Дж. Грея (руководителя исследовательского отдела фирмы Микрософт) о состоянии, эволюции и перспективах развития компьютинга стимулировали активные исследования не только в области постбинарного компьютинга, но также и исследования закономерностей и

перспектив развития компьютерных систем в существенно более широком контексте, что в итоге привело к формированию идей ноокомпьютинга [14–16] и целому ряду публикаций о закономерностях развития информационно-компьютерных технологий и техносферы в целом [17–27].

Под влиянием общения с другими известными профессорами Штутгартского университета (Р. Лаубер, П. Гёнер, М. Цайтц и др.) и участия в их научных семинарах во многом окончательно сформировались также и такие направления исследований как универсальные моделирующие среды [28–31] и суперсенсорный компьютинг [32–36]. При этом постепенно сформировалось следующее наблюдение: и универсальные моделирующие среды, и суперсенсорный компьютинг предполагают массовое накопление и обработку различных данных из окружающей среды, получаемых обычно с относительно ограниченной точностью, а это означает, что они не смогут в будущем эффективно развиваться без широкого использования постбинарного кодирования и постбинарных вычислений [37, 38].

Следует также отметить, что с 2007 года были также начаты исследования в области постбинарных клеточных автоматов, что позволило, с одной стороны, обеспечить наглядную демонстрацию особенностей и преимуществ постбинарного квазигенетического кодирования, а с другой – сформировать, фактически, новый класс клеточных автоматов, обладающих целым рядом существенно новых интересных свойств [39–46].

С 2010-го года началась весьма продуктивная совместная работа авторов данной монографии в области постбинарного компьютинга (с проведением еженедельных семинаров и привлечением целого ряда

аспирантов и магистрантов кафедры компьютерной инженерии ДонНТУ), что позволило резко интенсифицировать исследования и существенно повысить их результативность. Об этом, в частности, свидетельствуют две монографии, опубликованные в 2011-м [47] и 2012-м [48] годах, а также более 20-ти совместных публикаций, большинство ссылок на которые содержатся в последующих главах, так как именно эти работы стали основой данной монографии.

1.3. Постбинарная логика — основа постбинарного компьютинга

Логика в самом общем виде понимается как наука о формах, методах и законах интеллектуальной познавательной деятельности. Постбинарная логика в контексте данного исследования рассматривается как расширение булевой алгебры (алгебры логики) на базе использования расширенного логического пространства, первые представления о котором были сформулированы в начале 90-х годов (рис. 1.2).

Фактически основная идея заключалась в переходе от традиционного одномерного логического пространства, в рамках которого до сих пор происходило развитие компьютерной логики (рис. 1.3), к двумерному пространству, образуемыми ортогональными осями, соответствующих булевым значениям 0 и 1.

На этой основе к 1995 году сформировалось представление о некоем подобии периодической системы развития компьютерной логики (рис. 1.4), в которой была сделана первая попытка систематизации различных логических систем, используемых (или пригодных для использования) в вычислительной технике.

При этом в качестве простейшей может рассматриваться нульмерная монологика, оперирующая с единственным логическим значением 1, а более сложные логические системы формируются путем увеличения размерности логического пространства и количества используемых логических значений (вплоть до логического континуума).

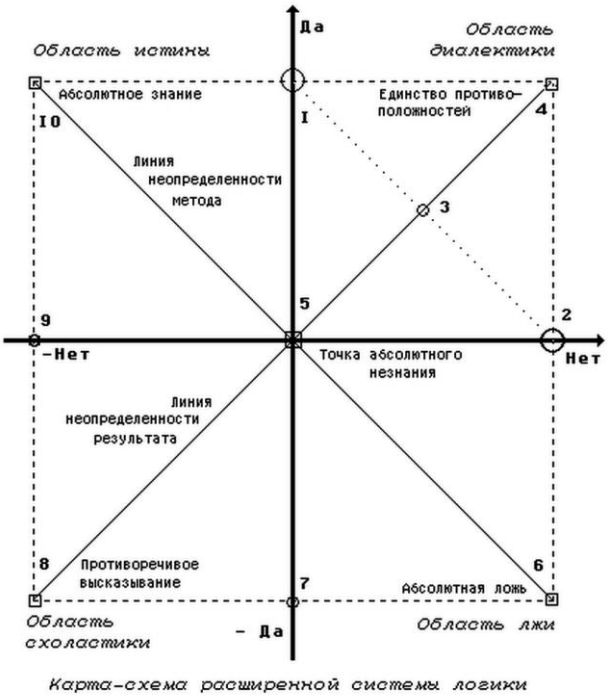


Рисунок 1.2 — Таким к 1992 году представлялось расширение традиционного двоичного одномерного логического пространства [10], при этом характерные точки просто пронумерованы: традиционному логическому значению «1» соответствует точка 1, а значению «0» – точка 2

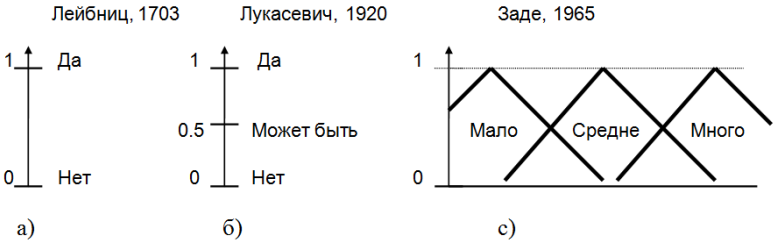


Рисунок 1.3 — Эволюция идей классической «одномерной» логики, предшествующей формированию и развитию многомерной постбинарной логики [11]: а) базовая бинарная логика; б) трёхзначная логика как простейший вариант многозначной; с) нечёткая логика, оперирующая функциями принадлежности.

Точка многозначности обозначена как A (от англ. «All» — «Все»), точка неопределенности обозначена как N (от англ. «Nothing» — «Ничего»). Верхний ряд соответствует существующим логикам в рамках традиционного одномерного логического пространства; средний ряд – соответствующие варианты логики в рамках предложенного двумерного логического пространства (в пределах «диалектического квадранта»). В нижнем ряду представлено полное двумерное пространство логики, включающее в себя кроме квадранта диалектической логики (DL) также квадранты схоластической логики (SL), позитивной логики (PL) и негативной логики (NL).

Важно также отметить, что в докладе, сделанном по результатам стажировки на институтском семинаре 31 января 1995 год были, в частности (в числе прочих вопросов), рассмотрены и перспективы развития «квазигенетической» логики и «квазигенетического» кодирования. При этом было предложено ввести в рассмотрение кроме традиционных логических и

арифметических операций специальные «генетические» операции, предполагающие генерацию конкретных «точечных» значений на основе «квазигенетических» кодов, а также – разного рода их модификации или «мутации».

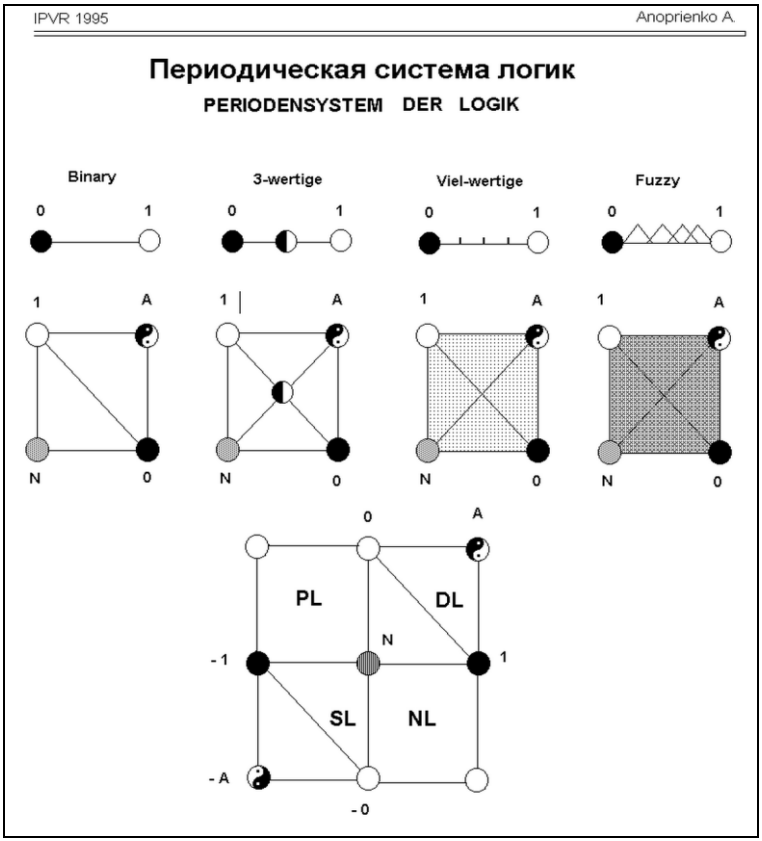


Рисунок 1.4 — Первый вариант «периодической системы логик», представленный на семинаре по итогам 4-месячной стажировки в Институте высокопроизводительных параллельных и распределенных компьютерных систем (IPVR) Штутгартского университета 31 января 1995 года

В процессе последующих исследований происходил процесс постепенного уточнения представлений о расширенном логическом пространстве и оптимизации системы обозначений (рис. 1.5 и 1.6).

В 1996 году в рассмотрение впервые вводятся понятия «тетралогика» и «тетракоды» [11], альтернативные определению «квазигенетические», что в последующем позволило обобщить данный подход как на кодо-логический базис, предшествующий бинарному («монологика» и «монокоды»), так и на базисы более высокого порядка, чем четверичные [49]. Особо следует отметить, что в работе [11] кроме двумерного логического пространства были также рассмотрены и двумерные интерпретации тетракодов.

Параллельно в этот период исследовались и вопросы возможного использования тетракодов для кодирования изображений [12, 13]. Кроме того, в контексте поиска средств и методов повышения эффективности вычислительного моделирования особое внимание уделялось развитию концепции универсальных моделирующих сред, ориентированных на сетевую (распределенную и/или параллельную) вычислительную инфраструктуру [29–31], что явилось, по сути, первым шагом к концепции расширенного алгоритмического базиса [50].

Важнейшим моментом при этом стало формулирование идеи о множественности эволюционирующих кодо-логических форм качественного и количественного представления знаний [49].

Утверждалось, в частности, что «**человеческий интеллект в зависимости от конкретной ситуации и решаемой задачи использует в процессе мышления не одну логическую систему, а некоторое достаточно**

представительное множество таких систем и связанных с ними количественных представлений. Традиционно используемая двоичная логика и основанные на ней системы счисления должны рассматриваться при этом в качестве одного из наиболее значимых, но отнюдь не единственных и не достаточных элементов современного интеллектуального инструментария. Другими важными составляющими являются как некоторые более ранние формы мышления и представления количественной информации, так и целый ряд перспективных, которые существуют пока только в зачаточном или не полностью оформившемся виде, но обладают значительным информационным потенциалом» [49, с. 59].

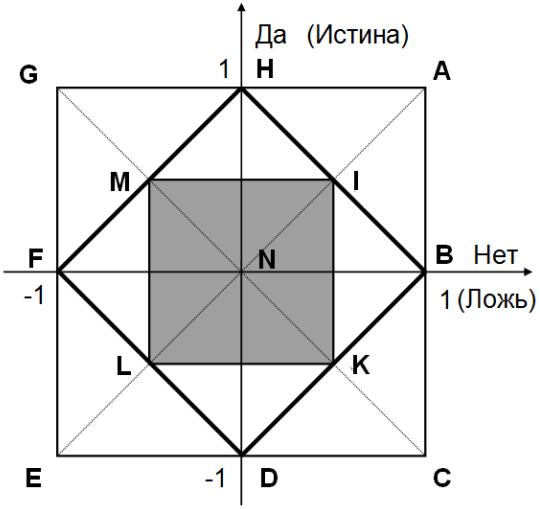


Рисунок 1.5 — Двумерное логическое пространство с ортогональными осями «Да» и «Нет» в варианте 1996 года [11]: характерные точки обозначены просто последовательными символа-ми латинского алфавита, начиная с A по ходу часовой стрелки (вначале внешние, а затем – внутренние)

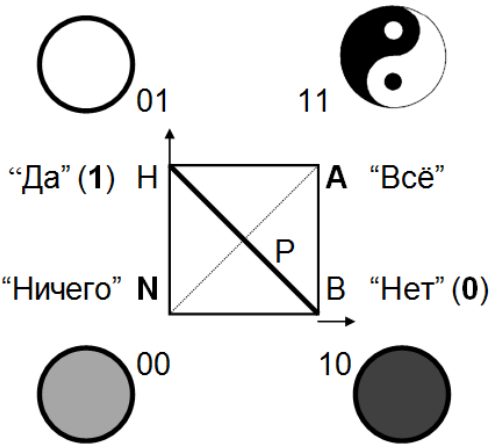


Рисунок 1.6 — «Диалектический квадрант» тетралогики (справа) в виде, сформированном к 1996 году: следует отметить актуальное и сегодня использование символики Инь-Ян в качестве иллюстрации сути значения множественности в тетралогике [11]

К 1997 году сформировалась окончательная система обозначений для так называемого «диалектического квадранта» – основного в контексте исторического развития логики и ее практического использования в настоящее время и в ближайшем будущем (рис. 1.7).

Модифицировалась также и система прочих обозначений с целью сделать их более логичными и осмысленными. При этом предполагался следующий смысл обозначений:

M — множественность, многозначность (и «истина» и «ложь», и да и нет одновременно).

A — обозначение полной неопределенности выбрано исходя из известной критики закона исключения третьего в «Науке логики» Гегеля: «Закон исключения третьего утверждает, что нет ничего такого, что не было бы ни А, ни не-А. Однако третье есть в самой этой тезе: само А есть третье, ибо оно может быть и +А и –А» [51, с. 482], т. е. значение его на момент высказывания утверждения не известно, и эта неизвестность и есть фактически тем самым третьим, обозначающим абсолютную неопределенность, «непроявленность», неизвестность.

S — симметричность (инверсная многозначность, отражение M относительно точки A).

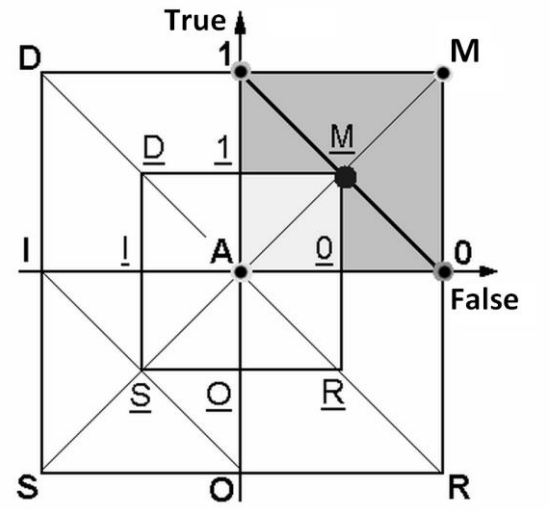


Рисунок 1.7 — Двумерное логическое пространство для различных вариантов тетралогики в виде, сформированном к 2005 году [8]. В большинстве обозначений характерных точек уже присутствует смысловая составляющая.

I и O — инверсные «истина» и «ложь» (обозначения были выбраны по подобию с 1 и 0, так как предполагается не только симметрия относительно точки A, но и относительно оси DR, при этом если 1 и 0 соответствуют положительному выбору некоторых значений из всего возможного множества, то I и O соответствуют отрицательному выбору, т. е. по принципу «все значения кроме данного»).

D и R — мнемонически соответствуют понятиям «дублирование» и «репликация», т. е. формы многозначности, по разному комбинирующие свойства значений M и S.

Каждому из перечисленных характерных значений может быть поставлено в соответствие значение, расположенное на середине расстояния между ней и A, обозначенные аналогичными символами, но с использованием подчеркивания, что мнемонически может ассоциироваться с дробностью, половинчатостью. Суть данных значений состоит в том, что в них неопределенность принимает вероятностный характер, т. е. равновероятны равноудаленные значения. Например, значение M предполагает равновероятность 0 и 1.

Приведенные обозначения существенно отличались от тех, которые первоначально использовались в 1996 году в работе [11]. Изменение обозначений были вызвано в основном двумя причинами: необходимостью улучшения их мнемонических свойств и стремлением к максимальному соответствию используемых обозначений (с учетом возможных их расшифровок) смысловому содержанию.

Смысл предложенных значений может быть проиллюстрирован на описанном ранее примере с бросанием монеты, но при условии, что мы описываем не

результат единичного случая, а поведение монеты в целом при проведении множества экспериментов. Если предположить, что классическая равновероятность выпадения орла или решки не является обязательной, то можно выделить следующие варианты знания о поведении монеты при бросании: A — ничего не известно и возможны любые варианты; M — монета ведет себя классически, обеспечивая равновероятность орла и решки; M — монета всегда при бросании падает на ребро и остается в вертикальном положении, оставляя одновременно открытыми и орла и решку; 1 — при бросании всегда выпадает решка; 0 — всегда орел; 1 — монета доступна для наблюдения после бросания только в половине случаев, при этом каждый раз наблюдается решка; 0 — аналогично предыдущему случаю, но наблюдается орел.

Таким образом, введение новых логических значений позволяет значительно расширить возможности формализованной логической оценки различных нюансов реальных процессов и ситуаций.

В 1997 году некоторые результаты исследований по многомерному кодо-логическому базису были впервые представлены в англоязычном информационном пространстве. В первую очередь речь идет о докладах на международной конференции по моделированию в Стамбуле [52, 53] и международном конгрессе по научным вычислениям, моделированию и прикладной математике в Берлине [54].

В 1999–2001 гг. удалось существенно продвинуться в исследовании и понимании феномена монокодов и монокодовых вычислительных моделей [55–58]. Это позволило не только понять многие закономерности эволюционного развития методов и средств кодирования

количественной информации, в т.ч. ее интеграции в комплексные вычислительные модели, но и впервые предложить достаточно обоснованные решения и интерпретации для некоторых достаточно известных трудноразрешимых научных проблем, среди которых в первую очередь можно назвать проблему Фестского диска [55].

В 2002 году возможности практического применения тетралогики были продемонстрированы путем разработки специальной методики модельной и компьютерной поддержки принятия решений в ситуациях когнитивного конфликта [59]. Применение предложенной методики и специальных средств ее реализации для всестороннего анализа одной из типичных научных проблем, характеризующихся обилием противоречивой и малодостоверной информации, позволило показать специфику и преимущества именно новых подходов к решению такого рода задач.

К 2005 году стало очевидным, что в дальнейшем двумерное логическое пространство может быть продуктивно расширено до трехмерного путем введения третьего измерения, соответствующего возможной недостоверности и/или «вариабельности» (т.е. возможной изменчивости) логических значений двумерного пространства (рис. 1.8) [8].

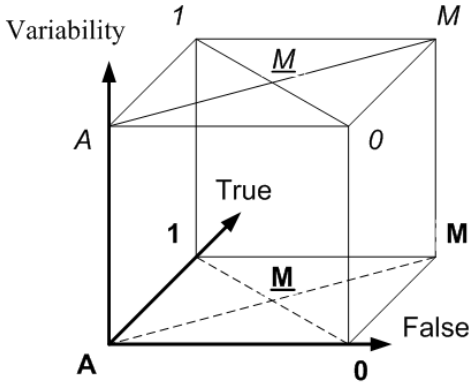


Рисунок 1.8 — Трехмерное логическое пространство с дополнительным измерением в виде вариабельности (Variability) логических значений

Трехмерное логическое пространство может быть порождено базисом, состоящим из ортонормированной системы векторов «Истина» (может обозначаться как T — «True» или Y — «Yes»), «Ложь» (F — «False» или N — «No») и «Вариабельность» (V — «Variability»). Логические значения при этом могут задаваться либо соответствующими координатами (например, в случае построения непрерывных логик), либо фиксацией характерных точек.

В трехмерном логическом пространстве дополнительно вводятся соответствующие значения *1*, *0*, *A*, *M* и *M*, модифицируемые при определенном условиях. Интерпретация такой модифицируемости может быть различная: например, адаптивность логики в смысле Д. Батенса [60] или допустимость ценностных изменений в логических значениях в контексте возможной смены или дополнения парадигмы «знания + аргументация» более гибкой парадигмой «знания + оправдания» [61, с.154].

В качестве одного из наиболее перспективных вариантов реализации логических систем в трехмерном базисе можно рассматривать октологику, которая в соответствии с введенной в работе [8] системой обозначений может быть описана как следующий кортеж значений:
LV8 = {0, 1, M, $\underline{M}$, *0*, *1*, *M*, *$\underline{M}$*}.

Следующим этапом в расширении размерности логического пространства может быть, например, дополни-тельная дифференциация источников возможной вариабельности значений, что приведет к логике с 16-ю основными логическими значениями. Дальнейший рост размерности с практической точки зрения представляется в обозримом будущем нецелесообразным.

С учетом всего, изложенного выше, на рисунке 1.9 представлена эволюция логического пространства с привязкой к исторической временной шкале.

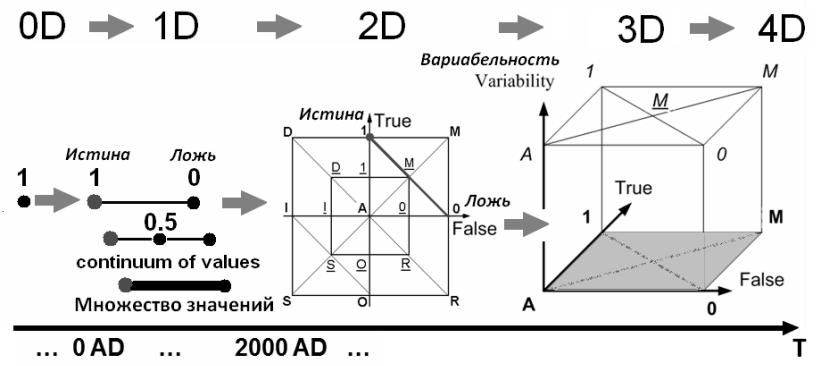


Рисунок 1.9 — Эволюция логического пространства от точечного «нульмерного» (слева) к многомерному (справа) с привязкой к шкале исторического времени (внизу):

Следует отметить, что исторически первым было нульмерное логическое пространство, основанное на использовании только значения «истина». Этот первичный вариант логики был обозначен как монологика и детально рассмотрен в целом ряде работ [56–58], а также в специальной монографии [62]. Переход к одномерному логическому пространству начался по мере использования символа «0» и завершился в основном к середине прошлого тысячелетия. Окончательно этот переход произошел на рубеже XVII и XVIII веков, благодаря, в первую очередь, работам Лейбница. Активное использование всего непрерывного одномерного логического пространства между значениями «0» и «1» началось только в XX веке.

Переход к двумерному логическому пространству актуализировался только на рубеже тысячелетий и, как предполагается, сможет оказать существенное влияние на развитие компьютерной логики в XXI веке.

1.4. Современная интерпретация расширенного логического пространства

Таким образом, первоначальные интерпретации и, соответственно, обозначения характерных логических значений в ходе развития концепции расширенного логического пространства подверглись существенным изменениям. При этом основной так называемый «диалектический квадрант» A1M0 не претерпел существенных изменений и его обозначения можно считать полностью устоявшимися как минимум с 2005 года. Но в интерпретации и обозначениях прочих квадрантов к настоящему времени сформировались

существенные уточнения, впервые описанные в 2013 году в работе [9] (рис. 1.10).

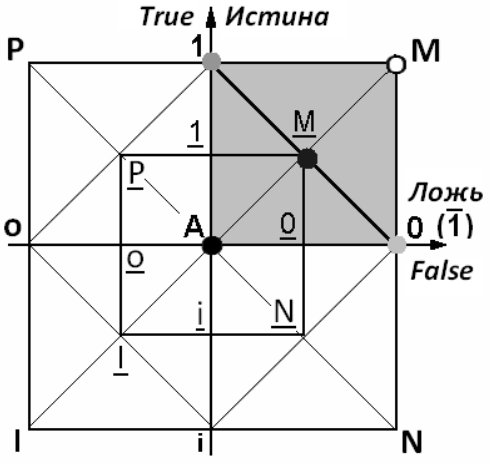


Рисунок 1.10 — Уточненное представление о полном двумерном логическом пространстве

В соответствии с этими уточнениями наиболее адекватными и естественными представляются следующие интерпретации и обозначения:

Квадрант AiIo является областью **воображаемой или «мнимой»** (по аналогии с «мнимой единицей» в исчислении комплексных переменных) логики. Это в основном соответствует первоначальной интерпретации данного квадранта как основанного на неявном (гипотетическом) получении знаний, а также с его определением как области схоластики, исходящей из тезиса первенства и главенства веры (т. е. сугубо умозрительной гипотезы) над разумом (т. е. осмыслением реальных фактов, проверенным экспериментально) [11]. К этой области может быть также отнесена воображаемая

логика Н.А. Васильева [63, 64] и многие разновидности современных паранепротиворечивых логик. В компьютерной логике к этой области целесообразно отнести правильные логические построения и конструкции, не привязанные к конкретным фактам и наблюдениям.

Квадрант A0P1 можно считать квадрантом позитивизма. Сюда, в частности можно отнести и современный неопозитивизм, или «логический позитивизм», в центре внимания которого находятся принципы эмпирической проверяемости научного знания. Но при этом, как правило, речь идет об учете только тех фактов, которые подтверждают логические утверждения и построения и опровергают противоречащие им факты и наблюдения.

Квадрант A0Ni можно считать квадрантом негативизма. Как противоположность позитивистского подхода данный вариант логики предполагает учет только опровергающих фактов или тех построений, которые отрицают подтверждающие факты. Этому квадранту соответствует, в частности, постпозитивизм К. Поппера, основанный на принципе фальсификационизма, согласно которому теория является научной, если существует методологическая возможность её опровержения путём постановки того или иного эксперимента, даже если такой эксперимент ещё не был поставлен.

Исходя из приведенной интерпретации квадрантов логического пространства можно выделить некоторые типичные **«траектории Познания»**, представленные на рисунке 1.11.

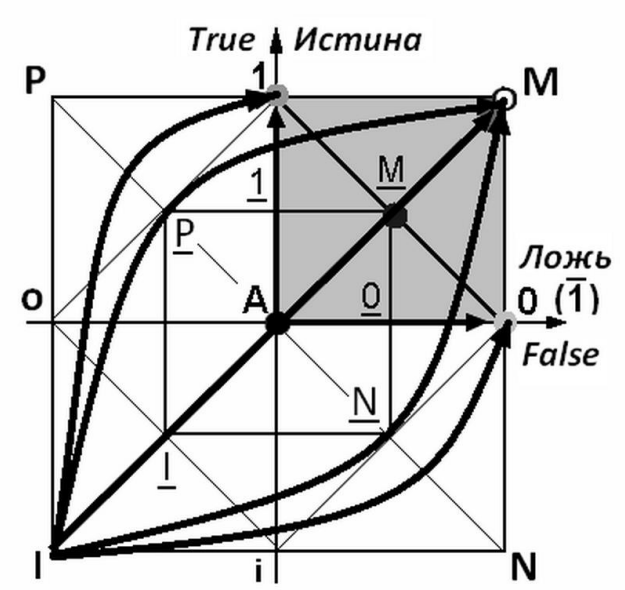


Рисунок 1.11 — Некоторые возможные «траектории Познания» в двумерном логическом пространстве

Среди этих траекторий можно выделить так называемые **«короткие»**, полностью лежащие в пределах диалектического квадранта и предполагающие эволюцию по мере накопления и анализа фактов от полного незнания (А) к полной определенности в виде значений 1, 0 или М (соответствующие траектории: А1, А0, АМ). Естественно, что кроме названных 3-х возможны и другие короткие траектории, имеющие более сложную форму и более неопределенный результат.

«Длинные траектории Познания» предполагают движение от некоторых полностью умозрительных гипотез в квадранте воображаемой логики к некоторым значениям в других квадрантах. При этом как сама траектория, так и

ее конечная точка во многом будет зависеть от используемых формулировок, отражая, в частности, известную истину, что законы объективны, а формулировки — субъективны. Как наиболее характерные можно рассмотреть следующие 5 длинных траекторий, начинающихся с точки I, соответствующей некоторой умозрительной гипотезе, включающей в себя гипотетические аргументы «за» и «против» (рис. 1.11):

I1 — простейшая траектория через осмысление возможных подтверждающих фактов, а затем – их выявление (верифицируемость) к полной истинности гипотезы;

IPM — более сложная траектория через сбор подтверждающих фактов и учет опровергающих факторов на заключительном этапе, что позволяет в итоге достичь понимания явления во всей его полноте;

IAM — траектория «диалектического движения» через точку полного незнания (через сократовское признание в определенный момент, что в действительности «я знаю только то, что ничего не знаю»), заключающееся в поиске и учете на всех этапах вначале самых разных, даже очевидно противоречивых, аргументов до точки признания полного незнания и/или непонимания, а затем и фактов как «за», так и «против» постепенно проясняющейся истины;

INM — траектория через сбор опровергающих фактов (фальсифицируемость по К. Попперу) и учет подтверждающих факторов на заключительном этапе;

I0 — траектория через сбор опровергающих фактов (фальсифицируемость) к полному отрицанию первоначальной гипотезы.

По-прежнему следует признать, что только диалектический квадрант представляет в обозримом

будущем практическую ценность как с точки зрения развития компьютерной логики, так и с точки зрения развития вычислительных возможностей различных компьютерных систем в контексте кодо-логической эволюции. Но с точки зрения развития систем искусственного интеллекта и компьютерной формализации всего наследия философской логики следует рассматривать все расширенное двумерное логическое пространство, не исключая в последующем и продуктивного использования третьего и четвертого его измерений.

1.5. Три основных потока развития логики

Анализ развития различных форм логики позволяет выделить три основных потока ее развития, существенно отличающихся терминологией, решаемыми задачами и результатами: компьютерный (вычислительный), математический и философский (рис. 1.12).

Компьютерная (вычислительная) логика в контексте нашего исследования может быть признана первичной, так как именно она в первую очередь обеспечивает практические вычислительные и алгоритмические потребности. При этом история вычислительной монологики и монокодов на сегодня может исчисляться десятками тысячелетий, но представлена она в основном множеством артефактов и/или традиций. Можно считать, что специальные исследования монологики как исторического явления и ранней стадии в развитии практической логики начались относительно недавно [56].

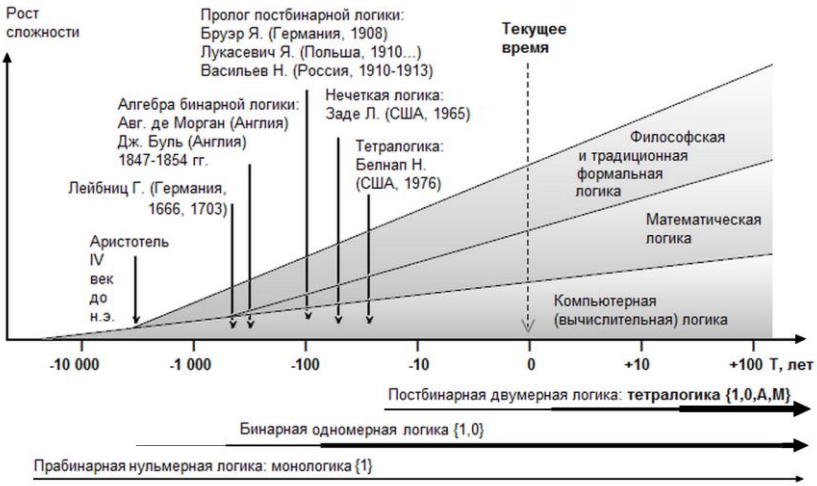


Рисунок 1.12 — Три потока развития различных видов логики (показаны выше оси времени), синхронизированные с тремя потоками (представленными ниже оси времени) и ключевыми вехами в развитии компьютерной (вычислительной) логики (шкала времени логарифмическая)

Философская логика в современном понимании начинается с работ Аристотеля, а математическая – с работ Лейбница и его современников, опубликованных преимущественно в середине XVII века и в самом начале XVIII столетия.

С работ Лейбница можно также вести отсчет и современной бинарной компьютерной логики и бинарного кодирования. В своём труде «Об искусстве комбинаторики», написанном еще в 1666 году, Лейбниц уже предвосхитил некоторые моменты современной математической логики. Двоичная система, практически в современном виде, была разработана им в первые годы

XVIII столетия и описана в работе Explication de l'Arithmétique Binaire, которая была опубликована в 1705 году. Характерно, что именно Лейбниц в конце XVII века создал один из самых первых механических калькуляторов, который выполнял сложение, вычитание, умножение и деление чисел, а также извлечение корней и возведение в степень на основе использования арабских цифр. Машина была продемонстрирована во Французской академии наук и Лондонском королевском обществе.

Важной вехой в развитии компьютерной логики стали выполненные в середине XIX века работы Джорджа Буля и Огастеса (Августа) де Мóргана. Джордж Буль был одним из первых математиков, вплотную занявшихся логической проблематикой. Идеи применения символического метода к логике впервые высказаны им в статье «Математический анализ логики» (1847). Последующие результаты были изложены им в обширном трактате «Исследование законов мышления, на которых основываются математические теории логики и вероятностей» (1854). В настоящее время так называемая булева алгебра является основной всей современной компьютерной логики. Август де Мóрган, шотландский математик и логик, профессор математики в Университетском колледже Лондона и первый президент Лондонского математического общества, к своим идеям в алгебре логики пришёл абсолютно независимо от Дж. Буля. В том же 1847 году, когда была опубликована статья «Математический анализ логики», он изложил свой взгляд на элементы логики высказываний и представил первую развитую систему алгебры отношений. С его именем связаны известные всем современным компьютерным специалистам законы де Моргана.

Развитие постбинарной логики фактически было инициировано в XX веке работами Я. Брауэра (1908) [65,

66], Н. Васильева (1910) [64] и Я. Лукасевича (1910, 1920) [67, 68], в которых в том или ином виде обосновывался отказ от принципа непротиворечивости и предлагалось использование третьего логического значения, суть которого выражалась словами «вероятно», «нейтрально» и т.п. В дальнейшем идеи многозначной логики активно развивались, но в основном в русле философской и математической логики. Великолепный обзор и анализ многочисленных исследований и результатов в этой области приведен в работе А.С. Карпенко «Развитие многозначной логики» [69].

Основным отличием тетралогики от различных вариантов философской и математической многозначной логики является не только ее четырехзначность и квазигенетический характер, но и неразрывная связь с тетракодами и арифметикой в контексте концепции единой кодо-логической эволюции информационных и компьютерных систем.

1.6. Кодо-логическая эволюция и ее основные составляющие

Условия для перехода к постбинарному компьютингу созрели как с точки зрения внутренней логики развития компьютерных систем и сетей, так и с точки зрения повышения эффективности их использования в самых разных областях применения. Важно также отметить, что такой переход представляется вполне естественным и закономерным в контексте кодо-логической эволюции. К настоящему времени уже вполне созрело осознание того, что следующим этапом развития компьютинга будет квазигенетическая эпоха на базе тетралогики и тетракодирования (рис. 1.13).

Рисунок 1.13 — К 2010 году квазигенетическое кодирование и квазигенетическая логика уже окончательно рассматривались в качестве основы будущего постбинарного квазигенетического компьютинга (слайд из презентации на международной конференции «Моделирование 2010» в ИПМЭ НАН Украины 13 мая 2010 года [70])

Следует также еще раз акцентировать внимание на том, что важнейшей особенностью проводимых авторами исследований является рассмотрение процесса развития компьютерной логики в тесной связи с эволюцией методов и средств представления количественной информации. Их неразрывная связь и взаимообусловленность позволяют рассматривать кодо-логическую эволюцию как целостный и непрерывный процесс, определивший развитие средств и методов компьютинга до начала нынешнего бинарного этапа и определяющий будущее постбинарное развитие.

На сегодня сформировалась достаточно целостная концепция кодо-логической эволюции, примерная структура которой представлена на рис. 1.14.

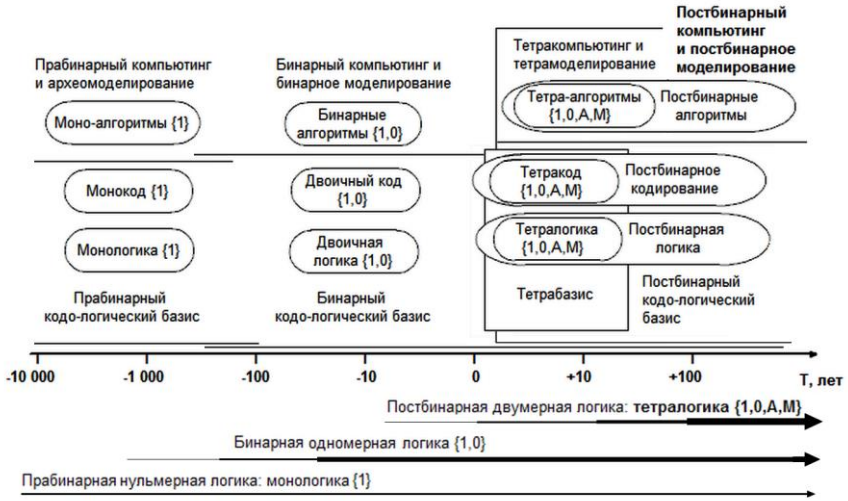


Рисунок 1.14 — Кодо-логическая эволюция: развитие и взаимосвязь основных составляющих, при этом шкала времени логарифмическая, в качестве нулевого значения обозначено текущее время, толщина линий в нижней части рисунка символизирует значимость различных вариантов компьютерной логики на разных этапах развития кодо-логического базиса

Алгоритмическая составляющих кодо-логической эволюции является одной из важнейших. В условиях интенсивного развития распределенных сетевых и многопроцессорных вычислительных структур все более остро проявляется существенная ограниченность возможностей современной бинарной логики и алгоритмов на ее основе в адекватном отражении всей сложности

реального мира. Практически уже назрела необходимость очередного качественного скачка в развитии логико-алгоритмического базиса вычислительных процессов [50], и в первую очередь это актуально для вычислительного моделирования, где имеющиеся ограничения ощущаются наиболее явно.

Аналогично развитию кодо-логического базиса в алгоритмической эволюции также могут быть выделены три основных этапа (рис. 1.15): моноалгоритмы (архаичный этап) на основе монологики и монокодов, бинарные алгоритмы (современный этап) на основе дилогики и дикодов, тетроалгоритмы на основе тетралогики (перспективный этап).

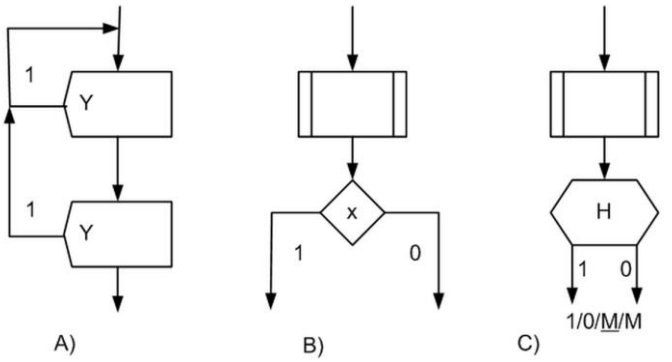


Рисунок 1.15 – Базовые концептуальные схемы моноалгоритма (A), бинарного алгоритма (B) и тетроалгоритма (C)

В работе [49] отмечалось, что в алгоритмическом плане монологике соответствует простая последовательность операторных вершин, выполнение которой реализуется в соответствии с простым правилом «если выполнен текущий оператор, то переходи к

выполнению следующего». Из логических операций с монологикой уверенно может быть связана лишь импликация (лат. Implicatio – сплетение), соответствующая связке «если..., то ...» и реализуемая в современных алгоритмических языках в виде условного оператора IF…THEN.

Типичный моноалгоритм носит инкрементный характер, заключающийся в последовательном переборе некоторого конечного счетного множества. При этом на каждом шаге может проверяться некоторое моноусловие Y, наличие которого ведет к изменению заданной последовательности шагов и переходу в некоторую заданную вершину алгоритма, которой, как правило, является начальная вершина.

Характерный пример реализации моноалгоритма показан на рисунке 1.16. Этот пример особенно интересен тем, что Мальтинская пластина, на которой он реализован, является на сегодня практически древнейшим (возраст порядка 20-ти тысяч лет!) из известных образцов вычислительных моделирующих сред, реализованных на основе монокода [56; 62]. Впервые детально вычислительная структура данной пластина была исследована в 1980-е годы профессором В.Е. Ларичевым [74, с. 198–261]. Аналогичные исследования данного артефакта активно продолжаются и в наше время, примером чего является работа [75].

Исследования Мальтинского пластины как специфического монокодового артефакта археомоделирования были выполнены в 1990-е годы [56] и позволили существенно расширить и уточнить результаты, полученные В.Е. Ларичевым и другими. В частности, были дополнительно выявлены и другие циклические процессы, отслеживаемые с помощью монокодовых узоров

пластины. Одним из наиболее интересных из числа этих процессов с алгоритмической точки зрения оказался 28-дневный женский репродуктивный цикл, длительность которого может существенно варьировать в зависимости от конкретной ситуации и индивидуальных особенностей.

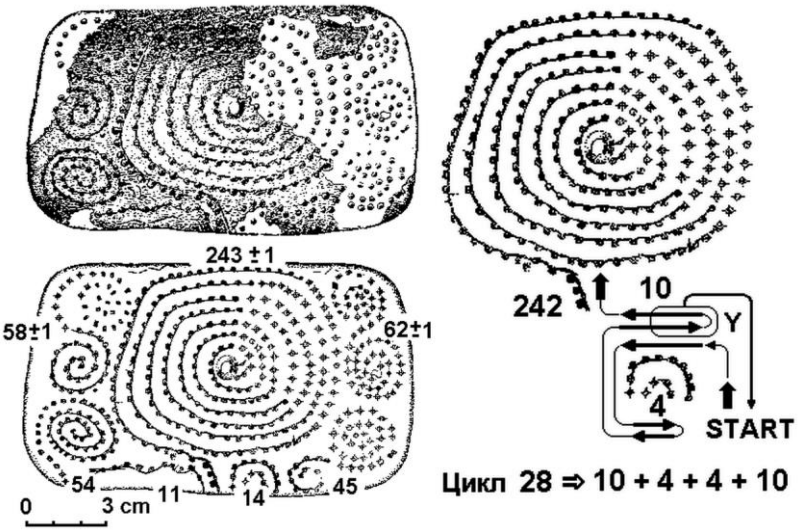


Рисунок 1.16 – Мальтинская пластина – древнейший (возраст около 20-ти тысяч лет!) из известных на сегодня примеров реализации монокодовых алгоритмов: в левой части – внешний вид верхней части пластины и ее модельно-вычислительная структура, справа – реализация типичного монокодового алгоритма отслеживания женского репродуктивного цикла [56]

Учесть эту вариабельность позволяет возможность немедленного перехода в начальное состояние START при наступлении события Y, означающего окончание очередного цикла и начало следующего. Так как

вариабельность цикла находится в пределах примерно 5-ти дней в сторону его сокращения или удлинения, то условие Y учитывается и действует фактически только в правой части участка «10». Регулярное наступление события Y является основным условием цикличности алгоритма, и в случае его отсутствия, означающего факт зачатия, алгоритм «по умолчанию» переходит в длинную спиральную часть «242» монокодового узора, которая завершается в центре сквозным отверстием, символизирующим рождение.

В целом, для моноалгоритмов значимым является только наличие (или появление) некоторого условия. Факт же отсутствия соответствующего условия специально не проверяется и приводит к реализации алгоритма в варианте «по умолчанию».

В бинарных алгоритмах после проверки в условной вершине некоторого условия осуществляется выбор одной (и только одной!) из ветвей перехода к дальнейшей реализации алгоритма.

В постбинарных алгоритмах (начиная с тетроалгоритмов — рис. 1.15 *C*) в условной вершине кроме традиционного бинарного выбора может также реализовываться случайный выбор или распараллеливание алгоритма в случае множественного выбора.

В целом важно отметить, что все основные составляющие кодо-логической эволюции, развивающиеся в неразрывной связи (логика, кодирование количественной информации и алгоритмы), в совокупности определяют эволюцию средств и методов вычислительного моделирования и компьютинга в целом.

При этом троичный компьютинг может рассматриваться как первая стадия постбинарного

компьютинга, реализованная уже в XX веке, а тетракомпьютинг – как последующая стадия развития, практическая реализация которой состоится уже в XXI веке. Схема, представленная на рис. 1.14, является по сути существенно упрощенной моделью такой эволюции во времени, так как в действительности различные этапы и их составляющие существенно перекрываются и границы между ними во времени довольно размыты, а более ранние формы в остаточном виде продолжают использоваться и на всех последующих этапах развития.

О том, что переход к постбинарному компьютингу не будет ни одномоментным, ни даже просто относительно быстрым процессом, весьма убедительно свидетельствует история перехода от монокодового компьютинга к бинарному (цифровому), на которой имеет смысл остановиться чуть более детально. Растянулась история этого первого перехода на многие столетия.

1.7. Закономерности и динамика перехода от монокодового компьютинга к бинарному

Древнейшие из известных на сегодня записей позиционной десятичной системы зафиксированы в санскрите, где не только использовались близкие к современным европейским названия количественных обозначений (2 – «дви», 3 – «три» и т.д.), но и широко применялась развитая словесная система обозначения чисел с использованием различных слов, однозначно связанных с определенными количественными представлениями. Например, для обозначения нуля использовались такие понятия как «дыра», «пустота» и т.п.; для единицы – «Луна», «Солнце»; для двойки – «крылья», «глаза», «близнецы», для четверки – «стороны

света». Число 1024 с использованием этой системы записывалось как последовательность слов: «Луна – дыра – крылья – стороны света». С VI века до н. э. фиксируется использование для записи количественных значений обозначений «брахми», с отдельными знаками для цифр от 1 до 9 (рис. 1.17).

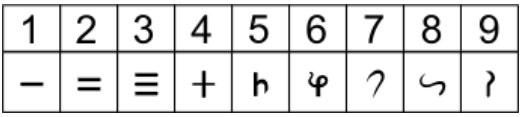


Рисунок 1.17 – Вариант начертания индийских цифр «брахми», характерный для I века н.э.: обозначения от одного до 3-х еще явно монокодовые, но в них уже угадываются современные цифры

К концу V века н. э. на базе использования цифр «брахми» сформировалась десятичная позиционная система записи чисел, которая с появлением в ней в последующие столетия специального обозначения для нуля в виде точки или кружка окончательно приблизилась к современной.

В дальнейшем индийская нумерация постепенно проникла сначала в арабские страны, а затем и в Западную Европу. В начале IX века Аль-Хорезми написал «Книгу об индийском счёте», способствовавшую популяризации арабских цифр и десятичной позиционной системы записи чисел во всём Халифате, вплоть до Мусульманской Испании. Арабский текст позже был утерян, однако сохранился его латинский перевод XII века «Algoritmi de numero Indorum». Но поскольку изначально труд аль-Хорезми был написан на арабском, то за индийской

нумерацией в Европе закрепилось неправильное название – «арабская».

Началом распространения арабских цифр в Европе можно считать 1202 год, когда Фибоначчи (Леонардо Пизанский) завершил свою «Книгу абака» (лат. Liber abaci), в которой описывалась и, фактически, пропагандировалась десятичная арифметика с использованием нуля. Именно Фибоначчи впервые вводит в Европе как самостоятельное число значение zero, т. е. ноль, название которого производит от слова zephirum, являющегося латинской формой арабского «ас-сифр», т. е. «пустой». От слова zephirum происходит также понятие «цифра», являющееся характерным атрибутом бинарного этапа развития компьютинга. С этого времени начинается весьма неспешный в свете нынешних представлений переход к цифровой эпохе. Но процесс этот не был непрерывным, а носил скорее волнообразный характер.

На рисунке 1.18 представлена концептуальная динамика первого этапа этого перехода, растянувшегося, по сути, на всю эпоху Возрождения. Основой для такого концептуального представления данной динамики явилась модель периодической составляющей системодинамики техносферы в виде последовательности волн Кондратьева или К-волн (К1, …, К6) с возрастающей амплитудой [20]. Постепенное нарастание амплитуды таких волн приводит к тому, что на гребне 5-й волны техносфера переходит на качественно новый уровень. В данном случае это переход к цифровой эпохе. В соответствии с данной моделью в последующие столетия предполагается некоторое затухание амплитуды К-волн, а затем — новая «раскачка» для перехода на следующий качественный уровень.

Следует отметить, что в истории этого периода развития цивилизации достаточно отчетливо

прослеживаются первая и вторая волна Проторенессанса (ПР1 и ПР2 — соответственно волны Возрождения В1.0 и В1.5), первая и вторая волна раннего Возрождения (РВ1 и РВ2 – соответственно В2.0 и В2.5), волна Высокого Возрождения (ВВ — В3.0) и волна Позднего Возрождения (ПВ – В3.5) [20].

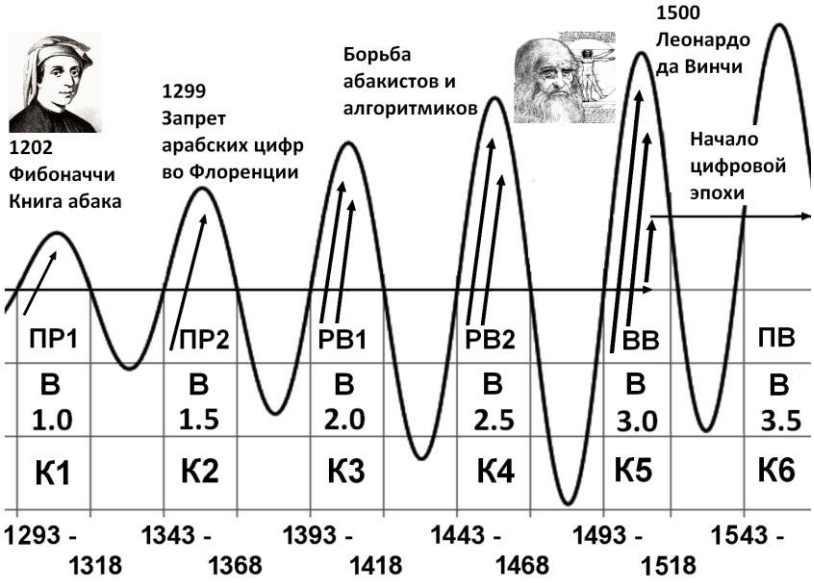


Рисунок 1.18 – Концептуальное представление динамики первого этапа перехода к цифровой эпохе в период Возрождения

Изучен данный период развития вычислительной техники пока явно недостаточно. Но постепенно нарастающие волны распространения цифровой записи можно проследить достаточно отчетливо. В частности, известно, что инициированные Фибоначчи первые прецеденты распространения в Европе десятичных цифр и арифметики на их основе, особенно в купеческой среде,

где требовались максимально эффективные и практичные инструменты счета, спровоцировали уже к концу XIII века интенсивную критику новой системы. Особенно жестко ее критиковали за то, что она весьма слабо защищена от возможных искажений: единицу, например, легко можно исправить на семерку, а приписать к числу лишние цифры – ещё проще. С традиционными монокодовыми числами, использующими, например, систему римского счета, такие махинации реализовать было существенно сложнее. Довольно широко известен факт официального запрещения арабских цифр в 1299 году во Флоренции, что стало следствием их «чрезмерного» на взгляд элиты того времени распространения.

Но постепенно достоинства индийских «арабских» становились все более очевидными для всех. Традиционно считалось, что уже к концу XIV века Европа почти полностью перешла на арабский цифровой код и пользуется им по сей день. Однако, при ближайшем рассмотрении оказывается, что это далеко не так.

В действительности с каждой новой волной прогресса усиливалась борьба так называемых абакистов, отстаивающих старую, монокодовую по своей сути, систему счета, и алгебраистов или алгоритмиков, пропагандирующих и продвигающих новую цифровую систему. Апогея эта борьба достигла к началу XVI века, что нашло свое отражение в многочисленных иллюстрациях того времени. Одной из наиболее характерных гравюр такого рода является иллюстрация из энциклопедического сочинения Грегора Рейша «Жемчужина философии» 1503 года, где абакисты персонифицированы Пифагором, олицетворяющим древнюю мудрость тысячелетий, на протяжении которых

использовался исключительно монокод, а алгебраисты – Боэцием (рис. 1.19).

Немного странным представляется то, что именно Боэций, живший на рубеже IV и V столетий олицетворяет на этой гравюре новую цифровую эпоху. Предположить в этой связи можно примерно следующее: во-первых, он был автором так называемых квадривиальных учебников — самых ранних его работ, датируемых 500-506 гг. и содержащих в числе 4-х дисциплин квадривия арифметику («Основы арифметики», традиционное латинское название — «Libri II de institutione arithmetica»); во-вторых, он перевел с греческого языка на латинский и сопроводил своими обширными комментариями важнейшие научные трактаты древности, в том числе главные логические труды Аристотеля. При этом переводы Боэция в большинстве случаев, по сути, превратились в толкования, содержащие оригинальные размышления, понятия и термины.

Объясняет он это следующим образом: «Сам я не следую покорно замыслам другого и не связываю сам себя строжайшим законом перевода, но, чуть свободней отклоняясь с чужого пути, ступаю не след в след. То, что у Никомаха говорится о числах пространно, я изложил с умеренной краткостью, а то, что изложено бегло и доходит до сознания с трудом, я раскрыл с помощью собственных скромных добавлений таким образом, что иногда для очевидности я пользовался и своими формулами и схемами» [71].

Возможно, именно поэтому Боэций стал символом новаторства в методах вычислений. Рядом с Пифагором он изображен и на знаменитой фреске Рафаэля «Афинская школа», завершенной в 1511 году (рис. 1.20).

Завершением первого этапа перехода к цифровой эпохе можно считать 1500-й год, когда в рукописях гениального Леонардо да Винчи появился рисунок, который принято сегодня считать эскизом первой цифровой вычислительной машины (рис. 1.21).

Рисунок 1.19 – Титульный лист Четвертой книги энциклопедического сочинения Грегора Рейша «Жемчужина философии» («Margarita Philosophica», 1503), посвящённой арифметике, изображает состязание двух великих учёных — Боэция-алгебраиста (слева) и Пифагора-абакиста (справа), судьёй в котором выступает сама Арифметика

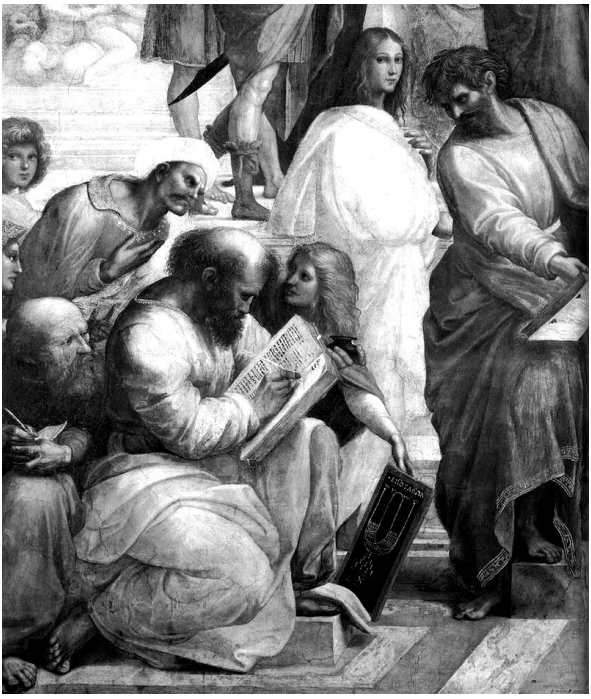


Рисунок 1.20 – Фрагмент фрески Рафаэля «Афинская школа» (1511 год): Боэций (крайний слева), подглядывающий в записи пишущего Пифагора

Но монокодовая эпоха на этом отнюдь не завершилась. Как показали современные исследования [72], индо-арабские цифры в европейских документах того времени вплоть до начала XVII века составляли всего лишь весьма малый процент от общего количества разного рода количественных обозначений.

Перелом в пользу цифровых технологий произошел только к середине XVII столетия (рис. 1.22), что практически немедленно привело к появлению первой действующей цифровой вычислительной машины,

реализованной Блезом Паскалем. В России, как и во многих других странах, до конца XVII века использовалась кириллическая система счета и лишь в начале XVIII века состоялся переход на арабские цифры.

Таким образом, потребовалось почти полтора столетия, чтобы от первых идей и эскизов Леонардо да Винчи перейти к реально работающим вычислительным устройствам. Идея уже долго буквально «витала в воздухе», о чем свидетельствует, например, обнаружение в архивных фондах библиотеки Штутгарта документов с описание конструкции цифровой счетной машины, предложенной еще в 1623 году профессором Тюбингенского университета В. Шиккардом [73, с. 60].

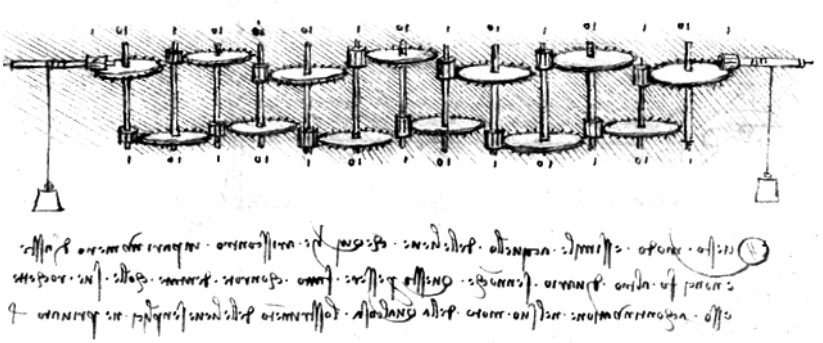


Рисунок 1.21 — Принято считать, что этот рисунок из записей Леонардо да Винчи, сделанных около 1500 года, является эскизом механизма первой цифровой вычислительной машины

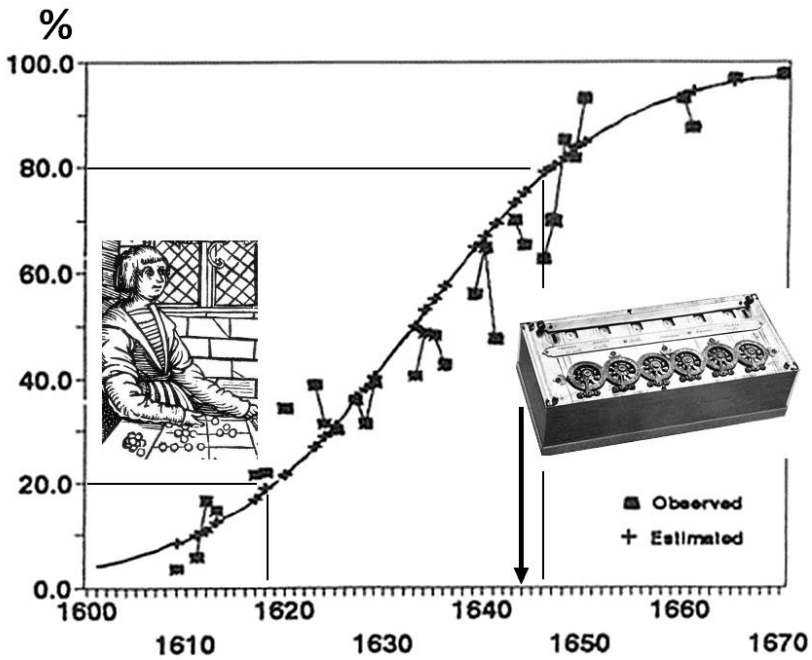


Рисунок 1.22 — Рост удельного веса цифровых обозначений в общем объеме количественной информации, содержащейся в англоязычных документах XVII века [72]: начало столетия характеризуется подавляющим доминированием различных форм монокода (на рисунке представлен характерный для того времени монокодовый «счет на линиях» – своеобразной форме абака), который окончательно уступает первенство цифровым технологиям лишь к середине столетия, что практически немедленно приводит к появлению первой действующей модели цифровой счетной суммирующей машины, созданной в 1642 году знаменитым французским ученым Блезом Паскалем.

Документально зафиксировано наличие первого реально работающего образца цифровой машины было лишь в 1645 году во Франции. Это была доведенная до уровня, пригодного для практического использования, машина Паскаля, сконструированная им еще в 1642 году в 19-летнем возрасте с целью облегчить работу своего отца, занятого подсчетом налоговых сборов. В письме отцу Паскаль писал о своем труде: «Я не экономил ни времени, ни труда, ни средств, чтобы довести ее до состояния быть тебе полезной... Я имел терпение сделать до 50 различных моделей: одни деревянные, другие из слоновой кости, из эбенового дерева, из меди…».

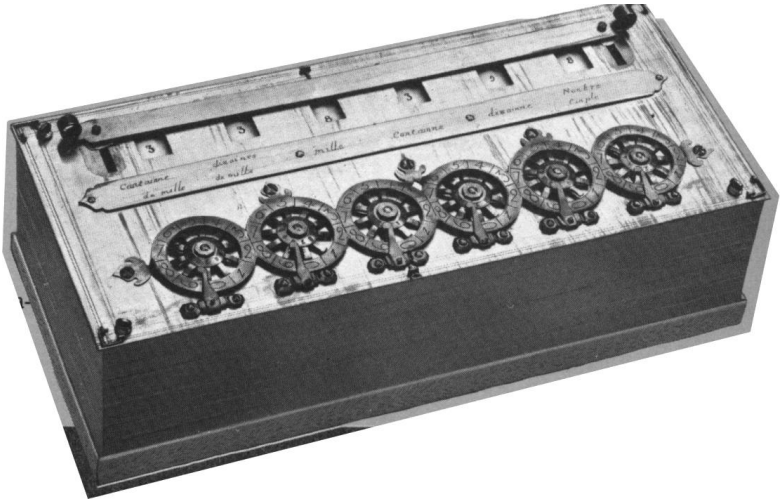


Рисунок 1.23 — Вычислительная машина Блеза Паскаля, открывшая через полтора столетия после первых эскизов Леонардо да Винчи эпоху практического использования цифровых вычислительных устройств

Современные историки вычислительной техники отмечают восторженную реакцию общества того времени на необычную новинку: «Машина Паскаля произвела на современников огромное впечатление. О ней слагались легенды и писались стихи. Множество людей приходило ее смотреть в Люксембургский дворец, где она была выставлена» [73, с. 68]. В 1649 году Блез Паскаль получил королевскую привилегию на счётную машину, в соответствии с которой категорически запрещалось как копирование модели Паскаля, так и создание без его разрешения любых других видов суммирующих машин. Запрещалась также их продажа иностранцами в пределах Франции. Все это существенно ограничило и задержало распространение вычислительных машин в Европе еще на полстолетия.

Но возможность практической реализации идеи цифровой вычислительной машины уже была доказана и не могла не вдохновить других изобретателей на создание новых образцов. Завершением волны появления первых практических полезных механических цифровых вычислителей в середине XVII века можно считать деятельность англичанина Самуэля Морленда. Лондонская газета в апреле 1666 года писала о его изобретениях следующее: «Сэр Сэмюэл Морленд изобрел два очень полезных инструмента: один служит для сложения и вычитания фунтов, шиллингов, пенсов и фартингов или любых других монет, весов и мер, другой — для быстрого выполнения умножения и деления, а также извлечения квадратного и кубического корней с любой требующейся точностью». Сохранилось три образца таких машин (рис. 1.24). Два из них находятся в Лондонском музее науки, один – в Оксфордском музее. В отличие от сложной и

дорогостоящей машины Паскаля эти устройства были относительно просты в изготовлении и использовании.

Рисунок 1.24 — Счетная машина Морленда, 1666 год

Решающую роль в подготовке широкого распространения цифровых вычислительных машин сыграл во второй половине XVII века Готфрид Лейбниц – основатель и первый президент Берлинской Академии наук. Лейбниц считается одним из самых всесторонних гениев за всю историю человечества. Отец кибернетики Норберт Винер утверждал, что после Лейбница уже не было человека, который бы полностью охватывал всю

интеллектуальную жизнь своего времени. И, по мнению того же Винера, если бы кибернетика нуждалась в святом покровителе, им должен был бы стать Лейбниц. Считается также, что в ходе своих неоднократных встреч с Петром I с 1697 по 1716 годы он существенно способствовал развитию научных исследований в России и созданию Академии наук в Петербурге.

Уже в 1666 году в возрасте 21 года он публикует работу «Об искусстве комбинаторики» («De arte kombinatoria»), где, опережая время почти на два столетия, предлагает проект математизации логики. Логику при этом он понимал как науку о всех возможных мирах. А ее основную задачу он видел в том, чтобы опираясь на очевидные «первые истины» логически вывести из них всю систему знания. Эта тема стала у Лейбница одной из ключевых и на протяжении всей жизни он разрабатывал принципы «универсальной науки», от которой, по его словам, «в наибольшей степени зависит благополучие человечества» [74, с. 480].

Счетная машина, над которой Лейбниц начал работать в 70-е годы, являлась, по сути, важной составляющей его работы по поиску «универсального языка». Первое описание «арифметического инструмента» было сделано Лейбницем уже в 1670 году. Окончательно идея вычислительной машины созрела у Лейбница после знакомства в 1672 году в Париже с другим великим математиком, физиком, изобретателем и астрономом того времени — Кристианом Гюйгенсом. В ходе своих астрономических изысканий Гюйгенс обнаружил туманность Ориона, описал кольца Сатурна и совершил много других открытий. При этом он был вынужден делать массу вычислений. Лейбниц, желая помочь коллеге в выполнении множества рутинных математических

операций, занялся практическим созданием своей вычислительной машины.

Первая публичная демонстрация «арифметического инструмента» состоялась в 1673 году на заседании Лондонского королевского общества. Лейбниц признавал определенное несовершенство нового прибора, но обещал его улучшить, чем с перерывами занимался на протяжении почти 40 лет своей жизни. В конце концов, примерно к 1694 году, он добился того, что на его калькуляторе можно было практически мгновенно перемножать 12-разрядные числа (рис. 1.25).

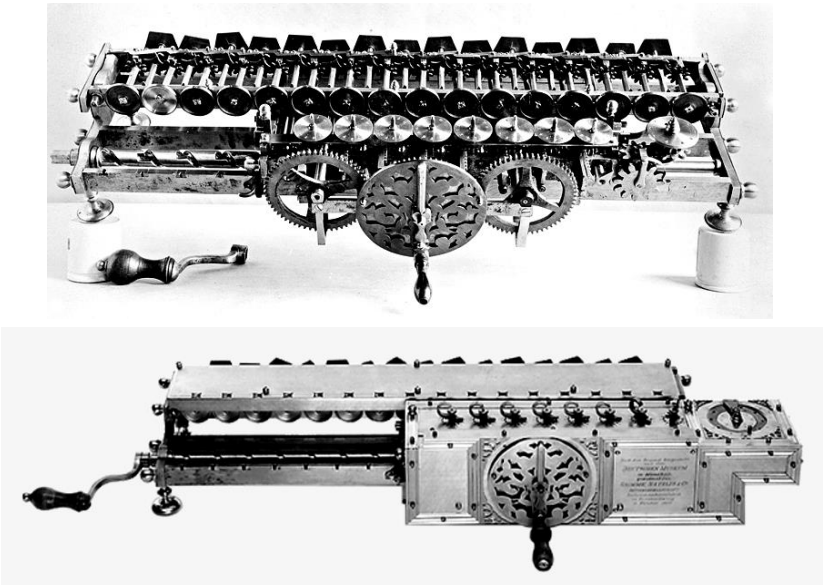


Рисунок 1.25 — Цифровая вычислительная машина Лейбница, ставшая к 1694 году (после многих лет разработки и совершенствования) фактически первым в мире арифмометром, открывшим эпоху механических вычислительных устройств (показаны современные модели-реплики калькулятора Лейбница)

В одном из своих писем он писал «Мне посчастливилось построить такую арифметическую машину, которая бесконечно отличается от машины Паскаля, так как дает возможность мгновенно выполнять умножение и деление над огромными числами» [73, с. 70]. Но и заплатить за реализацию своей идеи Лейбницу пришлось немало: общая сумма затрат была эквивалентна зарплате министра того времени почти за четверть века. Основными элементами машины Лейбница, позволяющими выполнять умножение и деление без последовательного сложения и вычитания, были так называемые ступенчатые валики, ставшие прообразом будущих зубчатых колес с переменным числом зубцов — основы арифмометров, массово выпускавшихся вплоть до середины XX века.

В 1697 году Лейбниц познакомился с Петром I. Первоначально их отношения были довольно прохладными. Лейбниц даже написал стихотворение, в котором желал побитому Петром Карлу XII завоевать Россию «от Москвы до Амура». Однако со временем между ними завязалась дружба, в результате которой российский император назначил ученому немалую пенсию и сделал тайным советником юстиции. В благодарность Лейбниц подарил Петру I экземпляр своего арифмометра, который (по неподтвержденным данным), Петр передарил китайскому императору.

Но главным в работах Лейбница было то, что параллельно с практической реализацией цифровой вычислительной машины, он впервые обнаружил, что любые цифровые позиционные системы счисления можно свести к двоичной системе, которая и является оптимальной для использования в вычислительной

технике. Детально двоичную арифметику он описал в трактате 1703 года (рисунок 1.26).

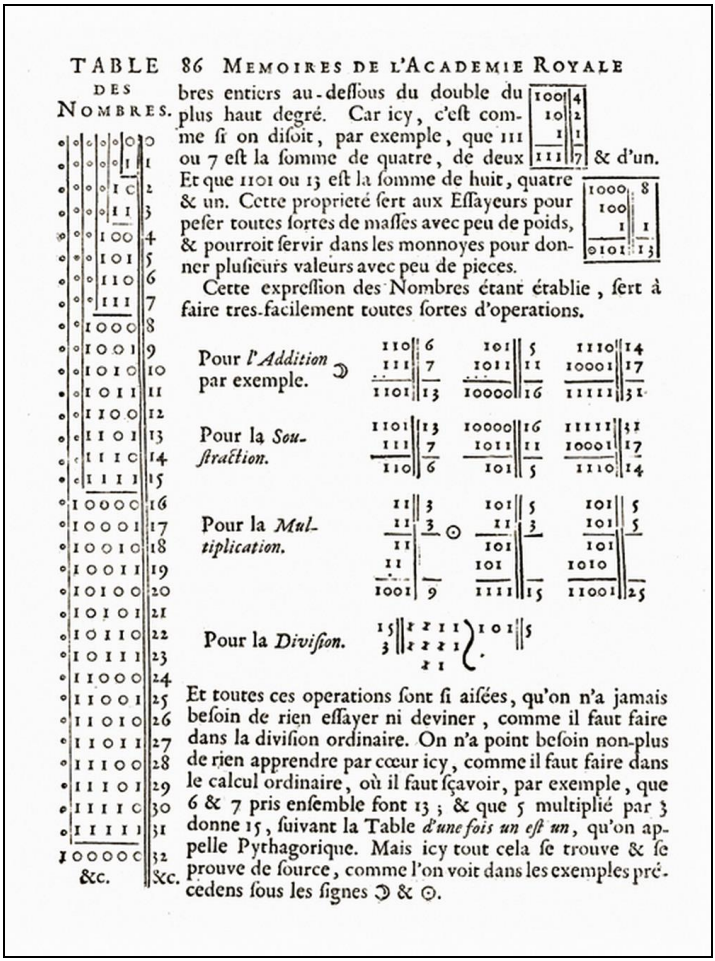


TABLE DES NOMBRES.

86 MEMOIRES DE L'ACADEMIE ROYALE

bres entiers au-dessous du double du plus haut degré. Car icy, c'est comme si on disoit, par exemple, que 111 ou 7 est la somme de quatre, de deux & d'un. Et que 1101 ou 13 est la somme de huit, quatre & un. Cette proprieté sert aux Essayeurs pour peser toutes sortes de masses avec peu de poids, & pourroit servir dans les monnoyes pour donner plusieurs valeurs avec peu de pieces.

Cette expression des Nombres étant établie, sert à faire tres-facilement toutes sortes d'operations.

Pour *l'Addition* par exemple.

Pour la *Soustraction.*

Pour la *Multiplication.*

Pour la *Division.*

Et toutes ces operations sont si aisées, qu'on n'a jamais besoin de rien essayer ni deviner, comme il faut faire dans la division ordinaire. On n'a point besoin non-plus de rien apprendre par cœur icy, comme il faut faire dans le calcul ordinaire, où il faut sçavoir, par exemple, que 6 & 7 pris ensemble font 13; & que 5 multiplié par 3 donne 15, suivant la Table *d'une fois un est un*, qu'on appelle Pythagorique. Mais icy tout cela se trouve & se prouve de source, comme l'on voit dans les exemples précedens sous les signes ☽ & ⊙.

Рисунок 1.26 – Фрагмент работы Лейбница по двоичной арифметике (Explication de l'Arithmétique Binaire), опубликованной в 1703 году в трудах французской академии наук (Memoires de l'Academie Royale des Sciences)

Обнаружению такого удивительного факта, что любые материальные понятия можно выразить цифрами «1» и «0» Лейбниц придавал настолько большое значение, что даже разработал специальную памятную медаль, посвященную «диадической системе» (рис. 1.27).

Рисунок 1.27 — Медаль «Образ Творения», разработанная Лейбницем в 1697 г., поясняющее соотношение между двоичной и десятичной системами исчисления и содержащая в верхней части девиз «Omnibus ex nihilo ducendis sufficitunum» (с лат. «Все из ничего, двойное образует единое»)

Лейбниц утверждал, что при сведении чисел к простейшим началам 0 и 1 везде появляется чудесный порядок. И. более того, Лейбниц считал, что двоичная

система — это одно из проявлений создателя, при этом цифра «1» — это «бог», а «0» — это «пустота», а из «1» и «0» произошло все».

И хотя о двоичной системе после Лейбница практически забыли до середины XX века, когда начали создаваться первые электронные цифровые машины, именно прозорливость Лейбница позволила осознать, что **с началом реализации цифровых вычислительных машин цивилизация перешла, фактически, к двоичной или бинарной эпохе**. Расцвет этой эпохи наступил с середины XX века в период НТР — научно-технической революции, а уже следующая волна получила название информационно-компьютерной революции (ИКР) и принесла осознание того, что **бинарный этап отнюдь не является завершением эволюции информационно-компьютерных технологий, а предполагает в будущем переход к следующему постбинарному этапу** (рис. 1.28).

В период первой промышленной революции на рубеже XVII и XIX веком именно машина Лейбница послужила образцом для многочисленных подражателей, разрабатывавших свои варианты вычислительных устройств. Итогом этого периода стало зарождение счетного машиностроения, впервые организованного талантливым инженером и предпринимателем из небольшого городка Кольмар в Эльзасе Карлом Ксавье Томасом (1785–1870), основателем и руководителем двух парижских страховых обществ с лирическими названиями «Феникс» и «Солейль» (Солнце). К 1818 году К. Томас на базе машины Лейбница создал для нужд своих страховых обществ собственную счетную машину для выполнения четырех арифметических действий и назвал ее арифмометром. Название «арифмометр», предложенное К. Томасом, прочно утвердился в счетной технике: все

машины, выполняющие четыре действия, с тех пор принято называть арифмометрами. К 1821 году в собственных мастерских в Париже К. Томас начинает уже серийное производство арифмометров: в первый год было изготовлено 15 машин, а затем ежегодно выпускалось до 100 экземпляров.

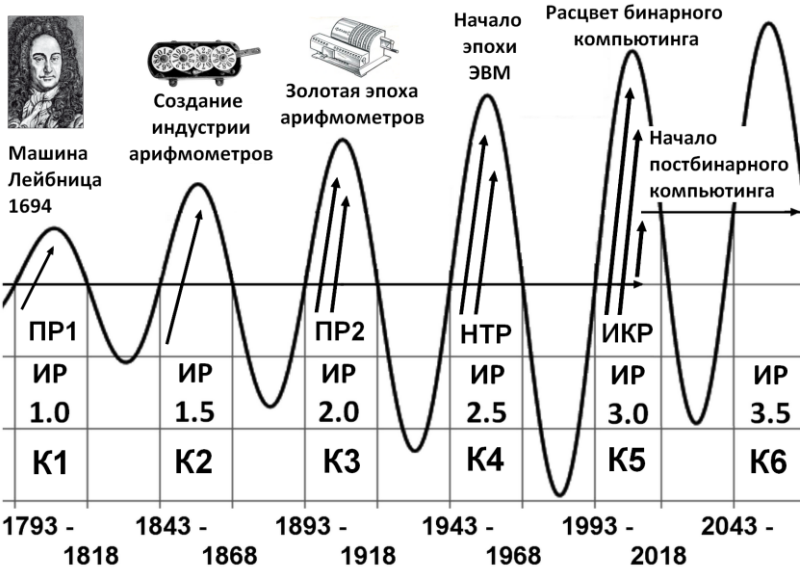


Рисунок 1.28 — Концептуальное представление динамики цифровой эпохи, где К1-К6 – классическая нумерация волн Кондратьева или К-волн, ИР 1.0–3.5 – волны индустриальных революций, ПР1 и ПР2 – первая и вторая промышленные революции в их классическом понимании

В период первой промышленной революции свершилось еще одно событие, которое в далеком будущем, в нашей современности, привело к массовому распространению программируемых компьютеров.

Традиционно считается, что самым первым программистом была помощница Чарльза Бэббиджа Ада Лавлейс. Но она была скорее программистом-теоретиком, т. к. машина, для которой она начала составлять свои первые в мире программы, так при ее жизни и не заработала. В действительности **первым в мире программистом-практиком следует считать французского изобретателя Жозеф-Мари Жаккара.**

В двадцать лет (после смерти отца) он наследовал в Лионе, столице ткачей Франции эпохи первой промышленной революции, ткацкое производство и стал думать над тем, как максимально механизировать труд ткача. Уже в 1790 году он предпринял первую попытку создать самодействующий ткацкий станок. В 1800 году он сумел усовершенствовать ручной ткацкий станок, создав приспособление для выработки крупноузорчатых тканей, и получил патент. На второй промышленной выставке в Париже в 1801 году на его модель удостоилась бронзовой медали. Следующей он изобрёл машину для вязания сетей и повёз её в 1804 году в Париж, где модели Жака де Вокансона, непревзойденного мастера механических игрушек, в том числе человекоподобных, вдохновили его на формирование окончательной конструкции станка, полностью завершённого только в 1808 году. Оценив в полной мере достоинства нового изобретения Наполеон I наградил Жаккара пенсией в 3000 франков и правом взимания премии в 50 франков с каждого действующего во Франции стана его конструкции. **Вскоре только во Франции работало более десяти тысяч таких станков.** Формирование сложнейших узоров на производимых с их помощью тканях программировалось перфорированными картонными картами (рис. 1.29).

Рисунок 1.29 — Портрет Жозефа Мари Жаккара, сотканный шелковыми нитями, для программирования которого потребовалось 24 тысячи перфокарт. Один экземпляр этого портрета был заказан Чарльзом Бэббиджем и, как считается, вдохновил его на использование перфокарт в его аналитической машине.

Фактически можно считать, что в перфокартах впервые для программирования использовалась двоичная система, в которой 2 возможных состояния кодировались наличием или отсутствием отверстия. Именно эти отверстия определяли порядок работы машин: когда карты проходили под специальными щупами и эти щупы попадали в отверстие, они опускались и с помощью особых устройств перемещали нити на ткацком станке, что позволяло формировать чрезвычайно сложные узоры.

Можно считать, что именно успех программируемых ткацких станков вдохновил Чарльза Бэббиджа, преподавателя кафедры математики Кембриджского университета, той кафедры, которую когда-то занимал Ньютон, на создание автоматических счетных машин с использованием перфокарт. Работу в этом направлении он начал в 1812 году, когда познакомился с первым опытом использования идей массового производство для математических целей, суть которого заключалась в том, что французское правительство разработало новый метод ускорения подсчёта математических и астрономических таблиц, основанный на принципах конвейера и параллелизма. Идея разделения труда вычислителей принадлежала Гаспару де Прони, руководившему бюро переписи Франции с 1790 по 1800 год. При этом предполагалось, что 3–4 квалифицированных математика определяют основные алгоритмы подсчётов, ещё примерно 10 математиков разбивают работу на более простые части, а сама рутинная работа, состоявшая из сложения и умножения, реализовывалась 80-ю работниками-счётчиками с довольно низкой квалификацией. Вычисления действительно удалось значительно ускорить, но платой за это стали многочисленные ошибки в результатах расчетов. Преодолеть это можно было только

путем использования машин, что исключило бы ошибки и позволило бы значительно быстрее выполнять расчеты.

В 1819 году Чарльз Бэббидж приступил к практическому созданию малой разностной машины для автоматизации вычислений. В 1822 году он закончил её разработку и выступил перед Королевским Астрономическим обществом с докладом о применении машинного механизма для вычисления астрономических и математических таблиц. Малая машина была полностью механической и использовала десятичную систему счисления. Она оперировала 18-разрядными числами с точностью до восьмого знака после запятой и обеспечивала вычисление 12-ти членов последовательности в минуту.

Первый успех вдохновил на создание большой разностной машины, которая позволила бы заменить большое количество людей, занимающихся вычислением различных астрономических, навигационных и математических таблиц, и наконец, полностью избавиться от ошибок, связанных с человеческим фактором.

С предложением профинансировать создание большой разностной машины Чарльз Бэббидж обратился в Королевское и Астрономическое общества. В 1823 году Бэббидж получил 1500 фунтов стерлингов и приступил к разработке новой машины, завершить которую планировалось за 3 года. Но Бэббидж недооценил сложность конструкции и существенно переоценил технические возможности того времени. Большая разностная машина должна была состоять из 25 000 деталей, весить почти 14 тонн, быть 2,5 метра высотой и иметь печатное устройство для вывода результатов. Память была рассчитана на тысячу 50-разрядных чисел. К 1827 году, когда сумма затрат уже более чем в 2 раза

превысила первоначальную сумму, ход работ по созданию разностной машины вначале сильно замедлился, а затем и полностью остановился. Реализовать в конечном итоге удалось лишь улучшенный проект, разработанный к концу 1840-х годов и названный «Разностная машина № 2» (англ. Difference Engine No. 2). Сделать это удалось шведскому издателю, изобретателю и переводчику Георгу Шутцу (швед. Georg Scheutz), который, основываясь на работах и советах Бэббиджа, начиная с 1854 года сумел построить несколько разностных машин и даже сумел продать одну из них в 1859 году канцелярии английского правительства. В 1855 году разностная машина Шутца получила золотую медаль Всемирной выставки в Париже. Спустя некоторое время другой изобретатель, Мартин Виберг (швед. Martin Wiberg), улучшил конструкцию машины Шутца и использовал её для расчёта и публикации печатных логарифмических таблиц.

Сам же Бэббидж с 1830-х годов переключился на создание **программируемой вычислительной машины, названной им аналитической**, ставшей основным делом его оставшейся жизни и принесшей ему посмертную славу. Аналитическая машина Бэббиджа должна была состоять из следующих частей: склада (store), фабрики или мельницы (mill), управляющего элемента (control) и устройства ввода-вывода информации. Склад предназначался для хранения значений переменных, с которыми производятся операции, и результатов операций, т. е. фактически это была в современном понимании память вычислительной машины. Мельница (фактически, арифметико-логическое устройство) должна была производить операции над переменными, а также хранить в регистрах значения переменных, с которыми в данный момент осуществляется операция. Третье устройство, которому Бэббидж не дал

названия (фактически это современные устройства управления или управляющие автоматы), осуществляло управление последовательностью операций, помещением переменных в склад и извлечением их из склада, а также выводом результатов. Оно считывало последовательность операций и переменные с перфокарт. Перфокарты были двух видов: операционные карты и карты переменных. Из операционных карт можно было составить библиотеку функций. Кроме того, по замыслу Бэббиджа, Аналитическая машина должна была содержать устройство печати и устройство вывода результатов на перфокарты для последующего использования.

Данная машина была уже фактически прямым прообразом будущих электронных вычислительных машин, являясь программируемой и двоичной. И **были все шансы реализовать ее в том или ином виде уже в середине XIX века на гребне новой волны индустриальной революции**. Тем более, что в 1843 году Адой Лавлейс, английским математиком и дочерью поэта Байрона, для машины Бэббиджа не только была написана первая в мире достаточно сложная программа вычисления чисел Бернулли, но и было сделано полное описании машины с анализом ее возможностей для решения различных вычислительных задач. Кроме этого, Лавлейс активно пропагандировала идеи Бэббиджа, проектировала некоторые узлы машины и исследовала вопросы эффективного применения двоичной системы счисления. Но в 1851 году Бэббидж отказался от дальнейшей работы» над реализацией машины, а в 1852 году преждевременно ушла из жизни и Ада Лавлейс. В итоге двоичные программируемые машины стали реальностью только через столетие, в середине XX века, будучи реализованными уже на электронной базе.

В то же время новая волна индустриальной революции середины XIX века привела к превращению зачаточного счетного машиностроения в целую отрасль индустрии. Началом этого процесса можно считать 1844 год, когда французский предприниматель Тома де Кольмар впервые представил свой арифмометр, разработанный им несколькими годами ранее на основе калькулятора Лейбница, на французской национальной выставке промышленных товаров. Успеха он особого не имел, но уже в 1849 году, на следующей выставке арифмометр был отмечен серебряной медалью. В 1851 году этот арифмометр наряду с аналогичными устройствами других разработчиков был представлен на первой Всемирной выставке в Лондоне. От других аналогичных машин он отличался тем, что являлся уже серийно производимым устройством, имеющим уникальный серийный номер и снабженным напечатанным руководством пользователя.

Фактически, Тома де Кольмар заложил в 1851 году основы целой индустрии арифмометров. Интенсивное совершенствование этого прибора, а также активное его продвижение (описание арифмометра было направлено всем коронованным и знатным особам Европы), привело к тому, что устройство не только было удостоено множества наград в период с 1851 по 1855 годы, но и начало выпускаться вначале десятками, а затем и сотнями экземпляров. Для Всемирной выставки в Париже 1855 года Тома построил гигантский двухметровый арифмометр, который занимал целый стол. К началу 1860-х годов количество изготовленных арифмометров, ставших, фактически, первым промышленным стандартом в вычислительной технике, стало исчисляться уже многими сотнями. В последующие десятилетия речь шла уже о тысячах и десятках тысяч устройств.

Символом золотой эпохи этого типа устройств в период с начала 1890-х годов до 1915 года (рис. 1.28), когда их производство было массовым вплоть до начала мировой войны, стал арифмометр Однера (рис. 1.30), разработанный российским механиком шведского происхождения В. Т. Однером.

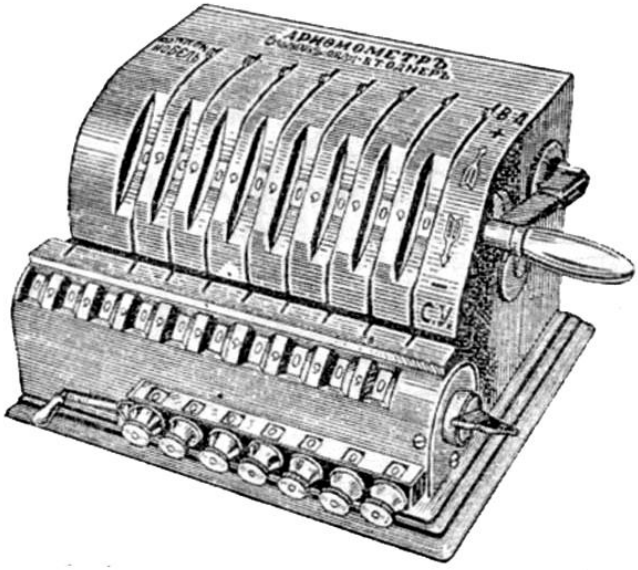


Рисунок 1.30 — Арифмометр Однера

Промышленное производство этих арифмометров впервые началось в Санкт-Петербурге в 1890 году. Уже с 1892 года начали появляться многочисленные клоны данного типа арифмометров, выпускавшиеся вплоть до второй половины XX века. В СССР, в частности, он выпускался почти полстолетия под названием «Феликс» в

1929-1978 годах. За этот период было выпущено несколько миллионов экземпляров этих машин.

Перелом в пользу электронных цифровых устройств произошел уже на гребне научно-технической революции середины XX века. Знаковым событием этого перелома можно считать появление на рынке в 1961 году первого полностью электронного калькулятора ANITA Mk VII. В связи с тем, что электронные компьютеры 1950-х годов стоили как минимум многие десятки тысяч долларов и имели огромные габариты, в английской компании Bell Punch Company было принято решение о создании электронной альтернативы для рядовых пользователей – настольного калькулятора, который стоил бы в сто раз меньше.

Секретный проект по производству электронного калькулятора получил кодовое имя ANITA, являющееся аббревиатурой слов «A New Inspiration to Accounting», что можно перевести примерно как «Вдыхающий новую жизнь в вычисления». Официально представлена ANITA Mk VII (рис. 1.31) была на Международной выставке бизнес-оборудования в Гамбурге в октябре 1961 года и сразу стала бестселлером, который десятками тысяч экземпляров начал быстро расходиться по всему миру.

Характерно, что в том же 1961 году сотрудники экспортного отдела Bell Punch Company повезли экземпляр ANITA Mk VII в Москву, где в рамках дней Великобритании с ней ознакомился Никита Хрущев (рис. 1.32). Но распространения в СССР в тот период данный калькулятор не получил, так как уже в 1964 году в СССР был разработан и начал серийно производиться первый советский полностью электронный калькулятор модели «Вега».

В то же время в СССР и на постсоветском пространстве практически до конца XX века наряду с электронными цифровыми устройствами продолжали довольно широко использоваться такие реликты монокодовой эпохи как деревянные счеты – прямые наследники древнего абака (рис. 1.33).

Рисунок 1.31 — Первый полностью электронный калькулятор для массового использования ANITA Mk VII, 1961 год

Важно отметить, что завершение эпохи вычислений на деревянных счетах отнюдь не означает уход в небытие всего наследия монокодовой эпохи. Некоторые реликты в виде, например, точечных монокодовых обозначений на игральных костях и костяшках домино с нами уже останутся навсегда (рис. 1.34).

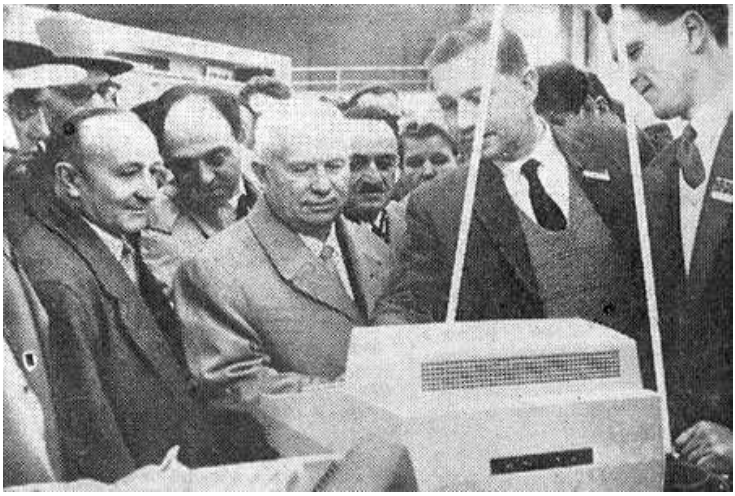


Рисунок 1.32 — Н. Хрущев и А. Микоян знакомятся с ANITA Mk VII, 1961 год

Рисунок 1.33 — Окончательный массовый переход от монокодовых по своей сути деревянных счет к цифровым калькуляторам бинарной эпохи произошел уже на памяти нынешнего поколения в конце XX века

Точно так же в будущем, когда, как предполагается, состоится массовый переход к постбинарным вычислениям, огромное наследие бинарной эпохи также никуда бесследно не исчезнет. В большинстве освоенных на сегодня областей применения, где нынешние возможности бинарной логики и обычной двоичной арифметики вполне достаточны, реликты бинарной эпохи также останутся в использовании на все обозримое будущее, но уже, скорее всего, как частный случай постбинарного компьютинга.

Рисунок 1.34 — А эти монокоды с нами уже, похоже, навсегда

Интересно также отметить, что как человеческий эмбрион в своем развитии ускоренно проходит все стадии эволюции, приведшей к появлению человека, так и в интеллектуальном становлении человеческого индивида на этапе первоначального обучения счету и вычислениям использование монокодовых вычислений на счетных палочках и счетах занимало, занимает и будет занимать важнейшее место. Характерно, что к началу нового тысячелетия, когда реальные вычисления на счетах ввиду массового распространения электронных устройств, практически повсеместно стали большой редкостью, в десятках стран мира начали распространяться специфические программы развития умственных способностей и творческого потенциала детей дошкольного и младшего школьного возраста с помощью арифметических вычислений на счетах, получившие название «ментальной арифметики» (рис. 1.35).

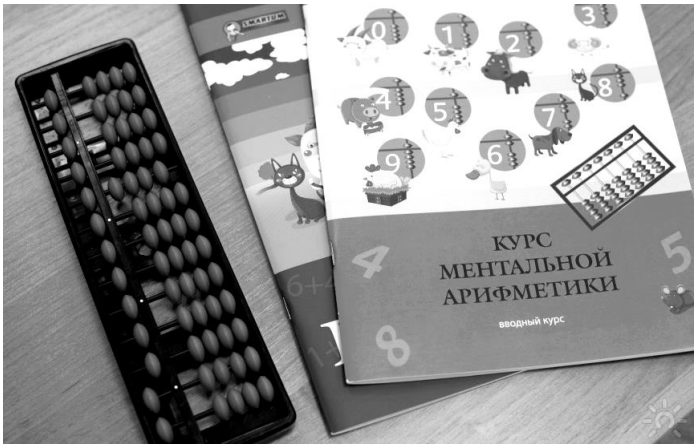


Рисунок 1.35 — В первоначальном обучении счету и арифметике монокоды также останутся уже навсегда

Считается, что так как цифры воспринимаются преимущественно аналитическим левым полушарием, а косточки абака – образным правым, то когда ребенок выполняет вычисления на абаке (счетах) или в уме, представляя абак, у него активно работает правое полушарие. Когда же ребенок получает задание или выдает ответ, он оперирует цифрами, и здесь уже задействовано левое полушарие. Таким образом, ментальная арифметика заставляет работать оба полушария вместе, образовывая прочные нейронные связи и способствуя гармоничному и всестороннему развитию интеллектуальных способностей. В заключение важно упомянуть о том, что способ вычислений с помощью счетов внесен в список устного и нематериального культурного наследия ЮНЕСКО.

1.8. Закономерности, определяющие условия перехода к постбинарному компьютингу

Как показывает анализ 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 (рис. 1.36). 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 приходится порядка 10-ти программируемых устройств, затем счет пойдет на сотни и

тысячи. Причем насыщение будет, скорее всего, происходить за счет наноустройств, встраиваемых во все мыслимые объекты техносферы, а также – в живые организмы, включая в первую очередь человека. Основная функция таких устройств будет заключаться в постоянном мониторинге и, при необходимости, регулировании и/или дорегулировании различных систем и процессов, начиная с наноуровня.

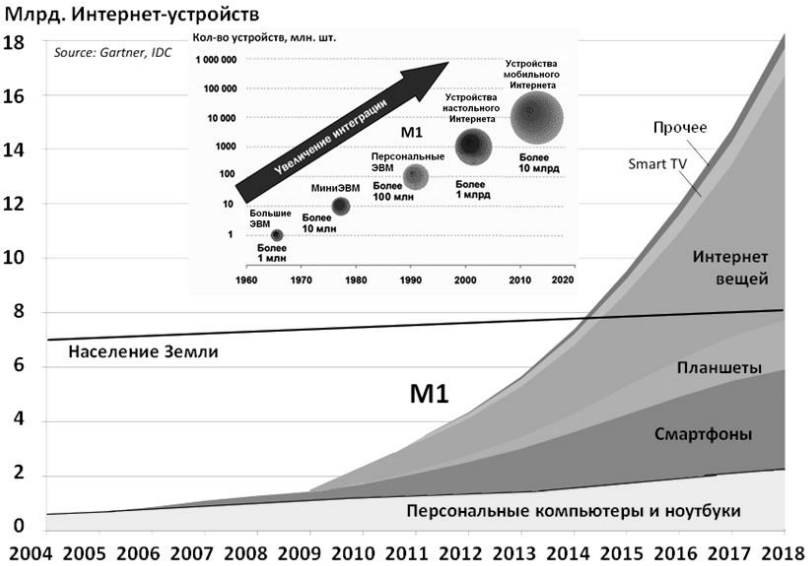


Рисунок 1.36 — Рост количества программируемых устройств: на порядок каждые 10 лет

Такое массовое использование программируемых систем, в том числе в жизненно важных приложения, например, для мониторинга и дорегулирования 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 (примерно в 2 раза) 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 (рис. 1.37 и 1.38).

В наиболее сложных современных микросхемах количество транзисторов уже превышает население планеты. Такой уровень сложности также предполагает необходимость дополнительных средств контроля достоверности и точности обработки информации, желательно уже на уровне представления данных, что может быть обеспечено переходом к постбинарному компьютингу. Удельные затраты на такой переход в пересчете на одно программируемое устройство окажутся чрезвычайно низкими, особенно с учетом того, что стоимость одного транзистора в составе микросхемы снижается особенно быстро – десятикратно каждые 5 лет (рис. 1.39).

Еще одним важным моментом является постоянное снижение проектных норм, а, следовательно, и размеров транзисторов в микросхемах (рис. 1.40), что привело в

настоящее время к необходимости реализовывать активные элементы размером менее 10 нм.

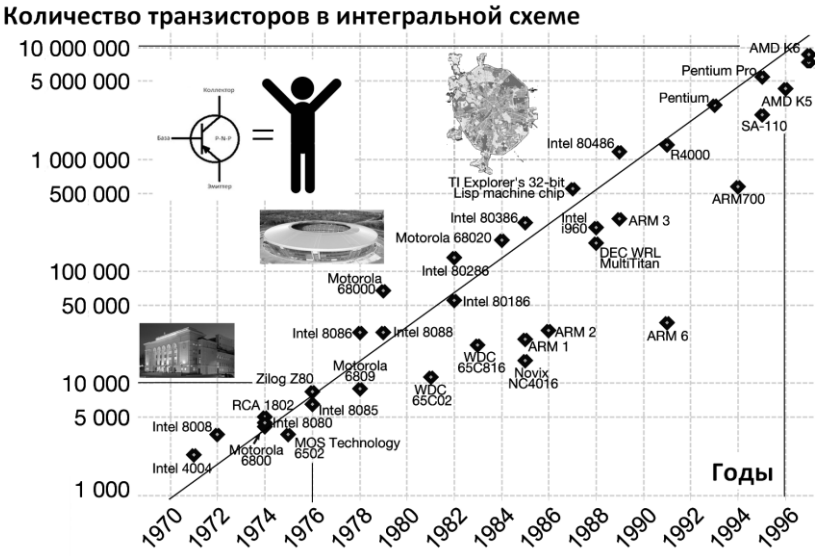


Рисунок 1.37 — Классический закон Мура в период с 1970-го по 1996 год: удвоение количества транзисторов в интегральной микросхеме каждые 18 месяцев или рост на 3 порядка каждые 20 лет (слева вверху показаны эквивалентные по масштабам объекты при условии приравнивания одного человека к одному активному электронному элементу: театр, стадион и город Москва)

При размерах электронных элементов в несколько нанометров и менее начинают активно проявляться различные квантовые эффекты, которые могут вносить дополнительные факторы неопределенности и ненадежности в функционирование электронных схем. Шансов на то, что в обозримом будущем удастся надежно

«обуздать» все эти квантовые явления и, более того, создать настоящие квантовые компьютеры, не так уж и много. Следовательно, надо предпринимать специальные дополнительные меры для преодоления нарастающей неопределенности и потенциальной ненадежности.

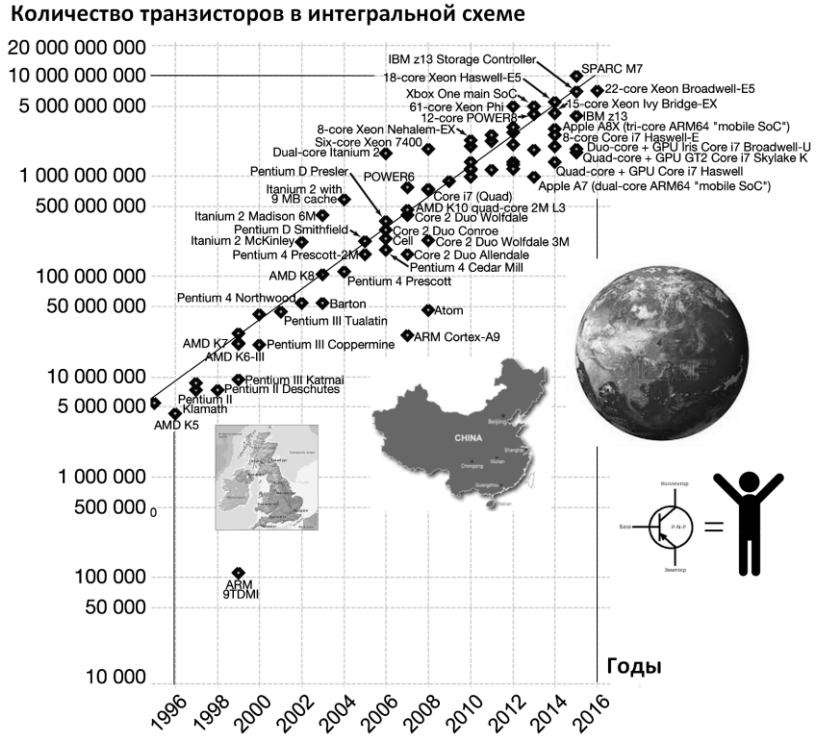


Рисунок 1.38 — Классический закон Мура в период с 1996-го по 2016 год: удвоение количества транзисторов в интегральной микросхеме каждые 18 месяцев или рост на 3 порядка каждые 20 лет (справа внизу показаны эквивалентные по масштабам объекты при условии приравнивания одного человека к одному активному электронному элементу: Британия, Китай, весь земной шар)

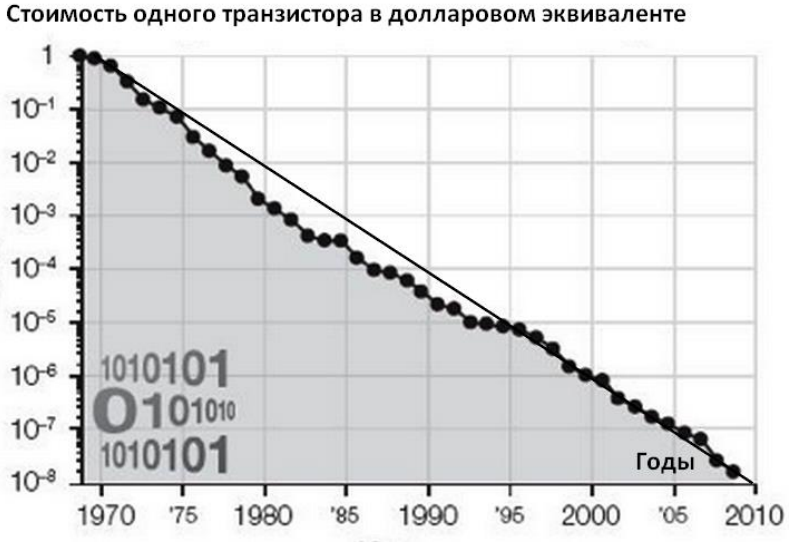


Рисунок 1.39 — Снижение стоимости одного транзистора в интегральных микросхемах: на порядок каждые 5 лет

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.

На рис. 1.41 представлена периодическая система роста производительности компьютерных систем различных классов от суперкомпьютерных (SCS. Классы 1 и 2) до нанокомпьютерных (классы 11 и 12) [21].

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.

Более того, оптимальным способом обеспечения наиболее эффективной реализации постбинарного кодологического базиса является как раз максимальная параллелизация обработки.

Если же попытаться сформировать наиболее вероятный сценарий перехода к постбинарному компьютингу, то следует предположить, что первые полностью постбинарные компьютерные системы будут реализованы при разработке новых классов устройств наноуровня, начиная с 12-го класса (рис. 1.41), так более ранние и масштабные классы систем довольно инерционны в своем развитии.

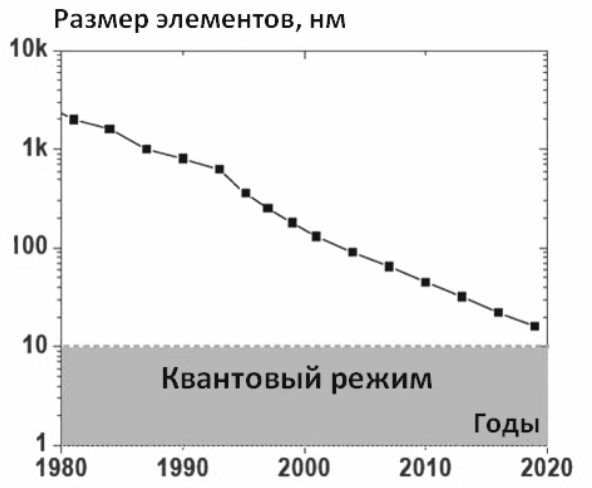


Рисунок 1.40 — Сокращение размеров транзисторов в интегральных микросхемах (на порядок каждые 20 лет) и приближение к квантовому режиму работы

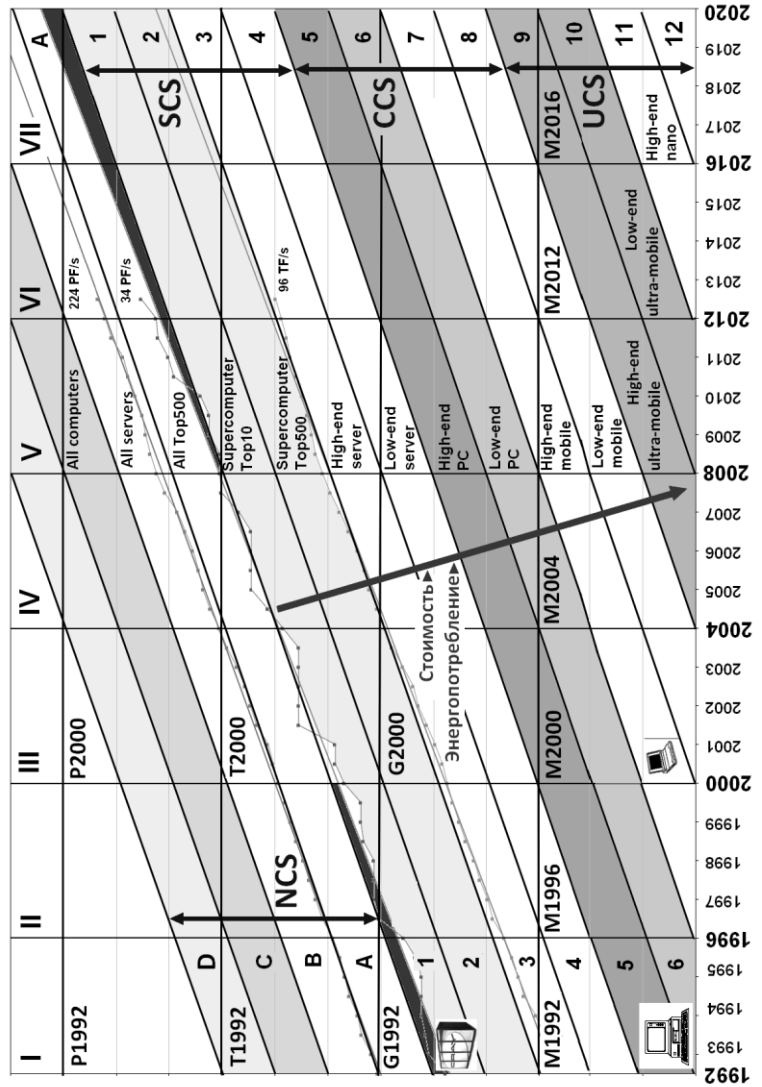


Рисунок 1.41 – Рост производительности различных классов компьютерных систем: на порядок каждые 4 года и каждые 4-8 лет появляется новый класс систем

1.9. Выводы

В целом следует отметить, что анализ растянувшегося на столетия, противоречивого и неравномерного перехода от монокодовой эпохи к бинарной отнюдь не обещает сколь-нибудь быстрого и беспроблемного перехода к постбинарному компьютингу. В настоящее время сложно прогнозировать конкретные сроки и особенности такого перехода. Можно лишь констатировать, что он уже начался и остановиться не сможет. Но его динамика, скорее всего, также будет зависеть в определенной степени от фаз кондратьевских волн, и, значит, следующий этап существенной активизации этого процесса может начаться где-то к середине XXI века.

А констатировать завершение процесса тотального перехода к постбинарному компьютингу придется, возможно, уже не в этом столетии. Это, естественно, не означает, что в ближайшие годы и десятилетия прогресс в данном направлении будет невозможен. Отнюдь нет! Всё в действительности будет зависеть от энтузиазма и приложенных усилий тех, кто заинтересуется идеями постбинарного компьютинга настолько, что своими исследованиями и разработками сможет их развивать и находить им все более широкий круг актуальных практических применений.

Глава 2
АРИФМЕТИКО-ЛОГИЧЕСКИЕ ОСНОВЫ ПОСТБИНАРНОГО КОМПЬЮТИНГА

В математической практике нет ничего более хлопотного, что более всего досаждало бы и мешало вычислителям, чем перемножение, деление или извлечение квадратных и кубических корней больших чисел. Выполнение этих операций не только приводит к значительным потерям времени, но и сопряжено с такой массой скрытых ошибок, что я начал размышлять о поисках надежного и удобного средства устранения подобных помех.

Джон Непер (1616 г.)

2.1. Основные принципы 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 [2]. 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 [5]. 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» [1, 6].

Необходимым условием при построении логической системы является выполнение главной задачи логики, которая базируется на двух ключевых моментах [1]:

- 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 соотношениях 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 вторая половина XX века ознаменовалась прогрессирующим нарастанием своеобразного протеста против двузначности, в частности:

- отвержение закона исключенного третьего и различные попытки преодоления «парадокса материальной импликации»;
- введение трехзначной логики Я. Лукасевича и общее усиление активности в области многозначных логик;
- разработка Н. П. Брусенцовым троичного ферритодиодного элемента и создание на его базе

троичной ЭВМ «Сетунь» в вычислительном центре МГУ [7, 8];

- появление нечеткой логики Л. Заде, справедливо квалифицируемой «как вызов, брошенный европейской культуре с ее дихотомическим видением мира в жестко разграничиваемой системе понятий» [9].

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 двумерное логическое пространство (рис. 2.1). 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); а также значения неопределенности (А) и множественности (М). 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 (вероятностных) логических состояний.

Состояния тетралогики могут кодироваться тетракодом, представленным в виде значений {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 [10], что в дальнейшем можно рассматривать как альтернативный (более узкий) вариант использования данного термина. Также, согласно [11], тернарная энтропия H_4 источника S с исходным алфавитом $\{a_1, a_2, a_3, a_4\}$ равна одному тетриту:

$$H_4(S) = -\sum_{i=1}^{4} p_i \log_4 p_i = \\ = -\sum_{i=1}^{4} \frac{1}{4} \log_4 \frac{1}{4} = -\log_4 \frac{1}{4} = 1 \text{ тетрит}, \tag{2.1}$$

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

Таким образом, тетралогика в простейшем варианте является конечнозначной и, согласно теории

функциональных систем [12], 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:

$$N_k^n = k^{(k^n)\cdot m}, \qquad (2.2)$$

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

Таким образом, согласно (2.2), 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 и имеют схематическое обозначение, представленное на рис. 2.1.

В последующем изложении 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.

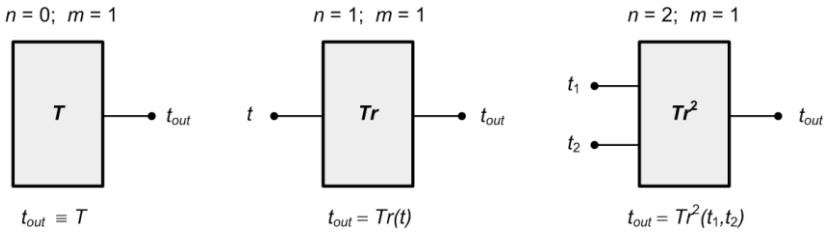


Рисунок 2.1 — Схематическое обозначение нольместных (нульарных), одноместных (унарных) и двуместных (бинарных) операций тетралогики

2.2. Тетралогические функции и операции тетралогики

Согласно (2.2) в тетралогике могут быть определены $N_4^0 = 4^{(4^0)\cdot 1} = 4^1 = 4$ простейшие нульарные тетрафункции, фактически являющиеся логическими константами (рис. 2.2):

- **логический тождественный ноль**, выражающий ложность события или логического состояния;
- **неопределенность А**, выражающая ни ложность ни истинность события или логического состояния;
- **множественность М**, выражающая одновременно и ложность и истинность события или логического состояния;
- **логическая тождественная единица**, выражающая истинность события или логического состояния.

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 $k = n = 4$: $\overline{A}_n^k = n^k = 4^4 = 256$.

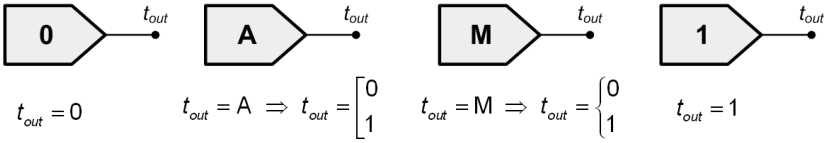


Рисунок 2.2 — Схематическое обозначение четырех логических констант

В монографии [3] определены и подробно изучены унарные тетрафункции. Там же [3, с. 68, табл. 2.1] приведены названия и обозначения 16-ти базовых унарных тетрафункций вместе с графической интерпретацией, которая повторена на рис. 2.3 и 2.4, некоторых базовых унарных тетрафункций для логического состояния x, определенного на двумерном логическом пространстве как множество $x \in L_4$.

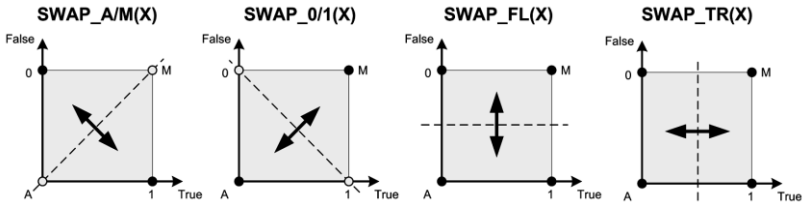


Рисунок 2.3 — Графическая интерпретация унарных логических инверсных операций SWAP

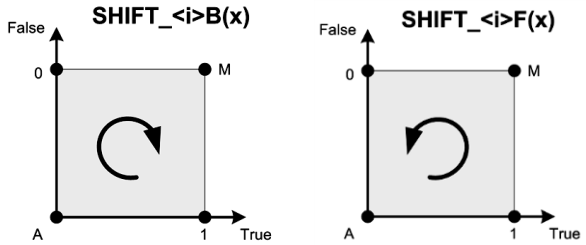


Рисунок 2.4 — Графическая интерпретация унарных логических сдвиговых операций SHIFT

Для **унарных логических операций инверсной группы** справедлив положенный в основу классической логики закон двойного отрицания [13], который в контексте тетралогики представляет явление, расширяющее не только концепцию двойного отрицания как такового, но и принадлежность к его 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:

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

Обратное утверждение

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

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…»* [3].

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

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

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 (рис. 2.4). Аналогичным образом под словом «вперед» (сдвиговая операция SHIFT_<*i*>F, F от англ. Forward — вперед) — направление сдвига, совпадающее с направлением оси «True». Целое положительное число <*i*> ($i \geq 0$) определяет количество сдвигов (или «поворотов» условной плоскости в концепции двумерного логического пространства).

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 (2.3), определяющее значение i как остаток от деления ω на 4:

$$i = \omega \bmod 4. \qquad (2.3)$$

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 (2.3), $i = 4 \bmod 4 = 0$, 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).

Тетрафункция отрицания или инверсии исходного логического состояния x может быть получена альтернативными тетрафункциями как сдвиговой, так и инверсной группы. Из табл. 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:

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

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:

$$\overline{x} \equiv \text{SWAP\_FL(SWAP\_TR(x))};$$

$$\overline{x} \equiv \text{SWAP\_0/1(SWAP\_A/M(x))}.$$

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 (рис. 2.5), а операция объединения ($\cup$) обладает свойством коммутативности.

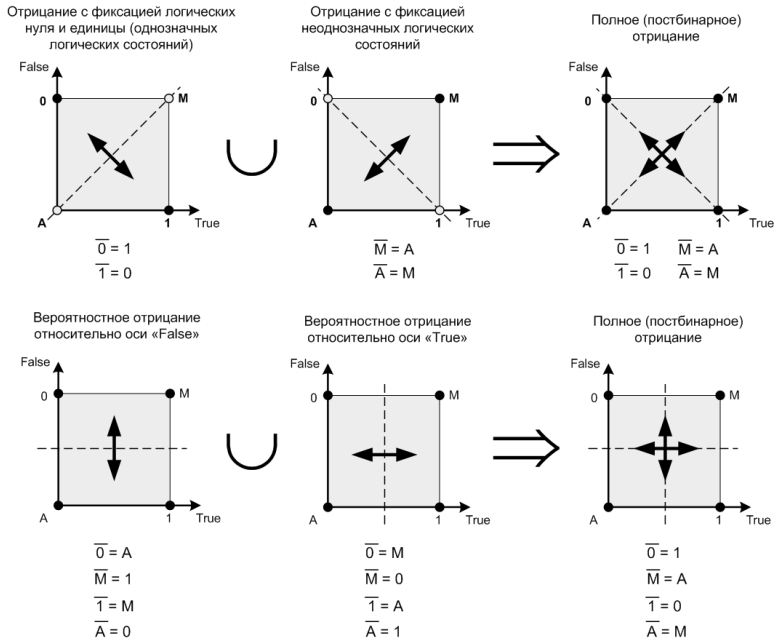


Рисунок 2.5 — Графическая интерпретация постбинарнго логического отрицания

Двуместные функции тетралогики позволяют создать множество полных систем функций представленной логики. Из (2.2) 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 тетрафункции 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 [4, 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}.

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:

$$\mathrm{A} \wedge 0 = 0;\ \ \mathrm{M} \wedge 0 = 0;\ \ \mathrm{A} \wedge 1 = \mathrm{A};\ \ \mathrm{M} \wedge 1 = \mathrm{M}.$$

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

$$\mathrm{A} \wedge \mathrm{A} = \mathrm{A};\ \ \mathrm{M} \wedge \mathrm{M} = \mathrm{M}.$$

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

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) [2, 14].

Решетка AG4 является простейшей диаграммой Хассе [15], в которой объединениями и пересечениями являются наименьшие верхние и наибольшие нижние

грани соответственно, а операция нестрогого включения ($\subseteq$) повышает логическое значение (рис. 2.6).

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.

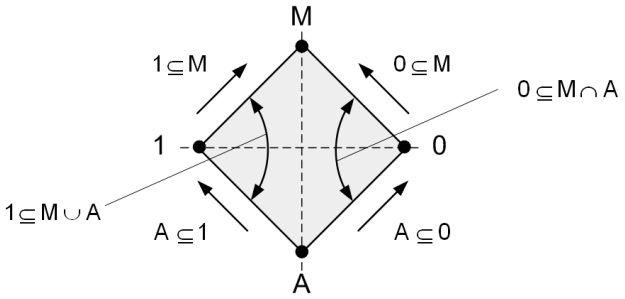


Рисунок 2.6 — Аппроксимационная решетка AG4

Теперь, поскольку $0 \subseteq \mathrm{M}$, по монотонности конъюнкции получаем $(0 \wedge \mathrm{A}) \subseteq (\mathrm{M} \wedge \mathrm{A})$, откуда, в силу свойства $x \wedge 0 = 0$, справедливо выражение

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

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

$$(\mathrm{M} \wedge \mathrm{A}) \subseteq 0. \qquad (2.5)$$

Связав соотношения (2.4) и (2.5) операцией конъюнкции и учитывая антисимметричность операции строгого включения, получим равенство

$$\big(0 \subseteq (\mathrm{M} \wedge \mathrm{A})\big) \wedge \big((\mathrm{M} \wedge \mathrm{A}) \subseteq 0\big) \Leftrightarrow \mathrm{M} \wedge \mathrm{A} = 0. \qquad (2.6)$$

Таким образом, благодаря соотношению (2.6), разрешилась неочевидность и парадоксальность выражения min(A, M), результатом которого является логический ноль: min(A, M) = 0. В табл. 2.1 сведены все сочетания конъюнкции тетралогики.

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.

Таблица 2.1 — Таблица истинности конъюнкции в тетралогике

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

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 (рис. 2.6). 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 шестнадцати 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 $\min(\mathrm{A}, \mathrm{M}) = 0$, то $\max(\mathrm{A}, \mathrm{M}) = 1$, следовательно $\mathrm{A} \vee \mathrm{M} = \mathrm{M} \vee \mathrm{A} = 1$.

2. Воспользуемся решеткой AG4 (рис. 2.6). Следует отметить, что логическое сложение эквивалентно операции объединения на решетке AG4. Поэтому поскольку $1 \subseteq \mathrm{M}$, то по монотонности конъюнкции получаем $(1 \vee \mathrm{A}) \subseteq (\mathrm{M} \vee \mathrm{A})$, откуда, в силу свойства $x \vee 1 = 1$, справедливо выражение $1 \subseteq (\mathrm{M} \vee \mathrm{A})$. Аналогичным образом $\mathrm{A} \subseteq 1$ приводит к $(\mathrm{M} \vee \mathrm{A}) \subseteq (\mathrm{M} \vee 1)$, откуда $(\mathrm{M} \vee \mathrm{A}) \subseteq 1$. Дальнейшие рассуждения вполне очевидны:

$$(1 \subseteq (\mathrm{M} \vee \mathrm{A})) \vee ((\mathrm{M} \vee \mathrm{A}) \subseteq 1) \Leftrightarrow \mathrm{M} \vee \mathrm{A} = 1.$$

В таблицу 2.2 сведены все сочетания дизъюнкции тетралогики.

Таблица 2.2 — Таблица истинности дизъюнкции в тетралогике

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

Авторами также рассмотрены **альтернативные способы определения конъюнкции и дизъюнкции** в тетралогике для любых сочетаний неопределенности и множественности.

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. рис. 2.2):

$$\text{А} = \begin{bmatrix} 0 \\ 1; \end{bmatrix} \qquad \text{М} = \begin{cases} 0 \\ 1. \end{cases}$$

Учитывая определения совокупности и системы выражений, очевидно получение следующих равенств:

$$\begin{bmatrix} 0 \\ \text{А} \end{bmatrix} = \text{А}; \qquad \begin{bmatrix} 1 \\ \text{А} \end{bmatrix} = 1; \qquad \begin{cases} 0 \\ \text{М} \end{cases} = 0; \qquad \begin{cases} 1 \\ \text{М} \end{cases} = \text{М}.$$

Таким образом, для конъюнкции справедливы следующие соотношения:

$$\text{А} \,\&\, \text{А} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \& \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ \text{А} \end{bmatrix} = \text{А}; \qquad \text{М} \,\&\, \text{М} = \begin{cases} 0 \\ 1 \end{cases} \& \begin{cases} 0 \\ 1 \end{cases} = \begin{cases} 0 \\ 1 \end{cases} = \text{М};$$

$$\text{М} \,\&\, \text{А} = \text{А} \,\&\, \text{М} = \begin{cases} 0 \,\&\, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ 1 \,\&\, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{cases} = \begin{cases} 0 \\ \begin{bmatrix} 0 = \begin{cases} 0 \\ \text{М} \end{cases} = 0. \\ 1 \end{bmatrix} \end{cases}$$

Для дизъюнкции справедливы следующие соотношения:

$$\text{А} \vee \text{А} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \vee \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \text{А}; \qquad \text{М} \vee \text{М} = \begin{cases} 0 \\ 1 \end{cases} \vee \begin{cases} 0 \\ 1 \end{cases} = \begin{cases} \text{М} \\ 1 \end{cases} = \text{М};$$

$$\text{М} \vee \text{А} = \text{А} \vee \text{М} = \begin{bmatrix} 0 \vee \begin{cases} 0 \\ 1 \end{cases} \\ 1 \vee \begin{cases} 0 \\ 1 \end{cases} \end{bmatrix} = \begin{bmatrix} \text{А} \\ 1 \end{bmatrix} = 1.$$

Результаты полученных выражений соответствуют значениям, приведенным в табл. 2.1 и 2.2.

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) [16, с. 157–166]. Определение конъюнкции и дизъюнкции в тетралогике с помощью аксиоматического аппарата теории множеств детально представлено в монографии [3, с. 77–93].

Таким образом, **предложенные операции отрицания, дизъюнкции и конъюнкции тетралогики сохранили в себе все важнейшие равносильности алгебры классической логики**. Это значит, что для логических значений, принадлежащих множеству {0, 1, A, M} справедливы законы коммутативности, ассоциативности и дистрибутивности дизъюнкции и конъюнкции. Ассоциативность дизъюнкции и конъюнкции позволяет опускать скобки в дизъюнкциях и конъюнкциях нескольких переменных, коммутативность — расставлять члены таких дизъюнкций и конъюнкций в любом порядке. Дистрибутивные (распределительные) законы дизъюнкции и конъюнкции 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).

Для тетралогики также справедливы законы де Моргана, которые позволяют выразить конъюнкцию через дизъюнкцию и отрицание, а дизъюнкцию — через конъюнкцию и отрицание.

Поскольку выполнение 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.

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

Аналогичным образом, в тетралогике с помощью базовых тетрафункций отрицания, конъюнкции и дизъюнкции можно вывести группу тетрафункций, эквивалентных булевым: $x \oplus y$ — сумма по модулю 2 (исключающее «ИЛИ); $x \downarrow y$ — отрицание дизъюнкции (стрелка Пирса — функция «ИЛИ-НЕ»); $x \leftrightarrow y$ — эквивалентность; $x \rightarrow y$, $x \leftarrow y$ — импликация (следование); $x \mid y$ — отрицание конъюнкции (штрих Шеффера — функция «И-НЕ»).

Ниже приведена таблица истинности обозначенных тетрафункций с указанием соответствующих преобразований (табл. 2.3). В силу новых исследований, проведенных авторами после 2011 года, данная таблица имеет незначительные изменения от ранее представленной в монографии [3, с. 92].

Таблица 2.3 — Таблица истинности тетрафункций

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
$x \oplus y$ $(\overline{x} \wedge y) \vee (x \wedge \overline{y})$	0	A	M	1	A	0	1	M	M	1	0	A	1	M	A	0
$x \downarrow y$ $\left(\overline{x \vee y}\right)$	1	M	A	0	M	M	0	0	A	0	A	0	0	0	0	0
$x \mid y$ $\left(\overline{x \wedge y}\right)$	1	1	1	1	1	M	1	M	1	1	A	A	1	M	A	0
$x \rightarrow y$ $\left(\overline{x} \vee y\right)$	1	1	1	1	M	1	M	1	A	A	1	1	0	A	M	1
$x \leftarrow y$ $\left(x \vee \overline{y}\right)$	1	M	A	0	1	1	A	A	1	M	1	M	1	1	1	1
$x \leftrightarrow y$ $(\overline{x} \vee y) \wedge (x \vee \overline{y})$	1	M	A	0	M	1	0	A	A	0	1	M	0	A	M	1

2.3. Основные принципы тетракодирования

Четверичная система кодирования количественной информации (**тетракод**) может быть определена различными комбинациями из четырех разрядов (**тетритов**), кодирующими состояния двумерного логического пространства в различных сочетаниях [6]. Однако тетракод $C_4 = \{0, \mathrm{A}, \mathrm{M}, 1\}$ получил более широкое применение в ряде исследований [17–19]. 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:

$$\tilde{A}_4^k = 4^k, \qquad (2.7)$$

где *k* — число разрядов тетракода (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, **постбинарные вычисления** — это компьютерные вычисления, оперирующие тетракодами, в которых содержатся числовые данные, а **постбинарное кодирование** — это тетракодирование, т. е. кодирование количественной информации с помощью тетракодов.

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 логические состояния;
- тетрит M приводится к битам 0 и 1 одновременно, поэтому представляется двумя точками на числовой оси;
- тетрит A приводится к битам 0 или 1 (в момент сведения его значение неизвестно), поэтому представляется одной из двух возможных точек на числовой оси.

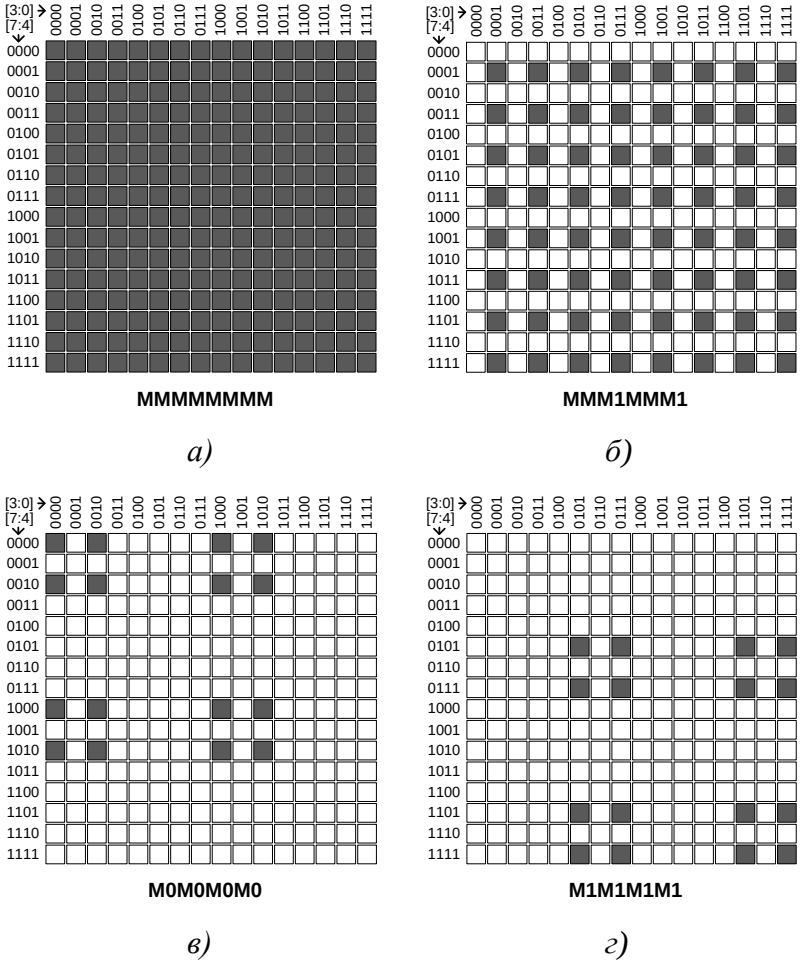
Тетриты 0 и 1 можно назвать **однозначно представимыми** на числовой оси, поскольку их значения сводятся к аналогичным двоичным значениям 0 и 1 в любой момент времени. Это подтверждено идентичностью

состояний «истина» и «ложь» тетралогики и классической бинарной логики.

Тетриты A и M являются **неоднозначно представимыми** на числовой оси, поскольку они не могут быть однозначно приведены к определенному двоичному значению. Тетрит A занимает одно случайное двоичное значение 0 или 1 в каждом событии (в определенный момент времени), т. е. он позиционируется одной из двух возможных точек на числовой оси: 0 или 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 (в любой момент времени), т. е. он всегда позиционируется одновременно двумя точками 0 и 1 на числовой оси. Аналогично, два тетрита M позиционируются четырьмя точками на числовой оси; три тетрита M — $2^3 = 8$ точками, а n тетритов M — 2^n точками на числовой оси.

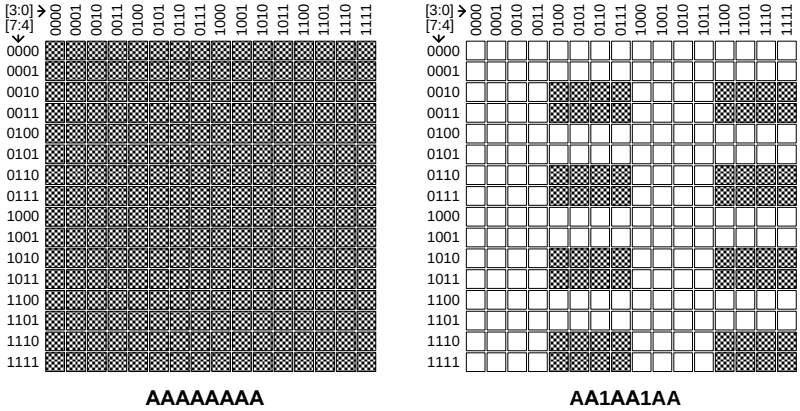
Графическая интерпретация приведения 8-разрядного тетракода к одному или совокупности двоичных кодов размерности [7:0] представлена на рис. 2.7, 2.8. При этом каждая ячейка определяет одно из чисел совокупности $0 \div (2^8 - 1)$ и пребывает в одном из трех состояний:

- □ — пустая ячейка: однозначно нет числа;
- ■ — заполненная ячейка: однозначно есть число;
- ▩ — вероятностная ячейка: число может быть или не быть с равной вероятностью.



а) *б)* *в)* *г)*

Рисунок 2.7 — Графическая интерпретация приведения 8-разрядного тетракода, содержащего разряды множественности М, к одному или совокупности двоичных кодов



а) *б)*

Рисунок 2.8 — Графическая интерпретация приведения 8-разрядного тетракода, содержащего разряды неопределенности А, к одному или совокупности двоичных кодов

На рис. 2.7 *а* тетракод MMMMMMMM приводится ко всем $2^8 = 256$ значениям, а на рис. 2.7 *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 рис. 2.7 *в-г* показано, как чередование 0, 1 и М позволяет осуществить выборку совокупности из требуемых $2^4 = 16$ значений. В частности, с помощью 8-разрядного тетракода MMMMMMM0 можно выделить все четные (а с помощью MMMMMMM1 — все нечетные) значения в диапазоне $0 \div 255$. На рис. 2.8 *а* тетракод AAAAAAAA приводится к одному из всех возможных $2^8 = 256$ значений, а на рис. 2.8 *б* — 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 A, однако они определяют область вероятной выборки.

На рис. 2.9 представлены результаты программного приведения тетракода к двоичному коду с последующим преобразованием к десятичному целому числу без знака. На основании полученных результатов можно сформулировать заключения, **определяющие основные принципы тетракдирования**:

1. При приведении *n*-разрядного 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 (рис. 2.9 *а*).
2. При приведении *n*-разрядного 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 битам 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:

$$\tilde{A}_m^n = \tilde{A}_2^n = 2^n, \qquad (2.8)$$

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 (рис. 2.9 *б*).

3. При приведении n-разрядного 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.

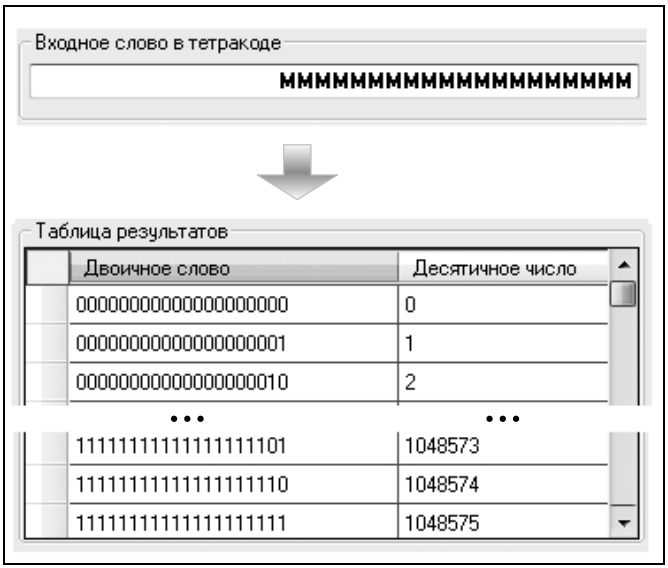
   При этом в результате повторного приведения такого тетракода может быть получен другой двоичный код с другим значением.

   Например, на рис. 2.10 *а–б* 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.

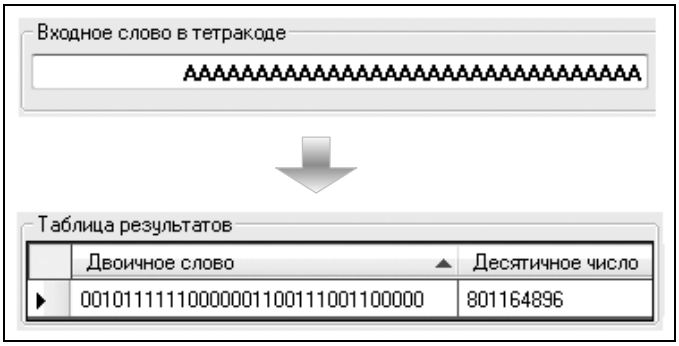
На рис. 2.11 представлены результаты приведения тетракода к двоичному коду — формату числа с плавающей запятой одинарной точности с последующим преобразованием к вещественному числу в десятичной системе счисления.

а)

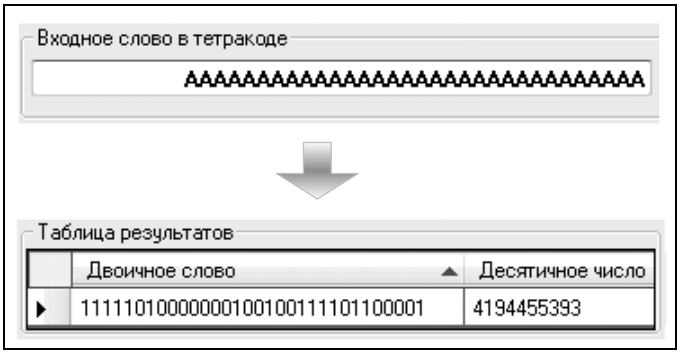


б)

Рисунок 2.9 — Результаты приведения тетракода к двоичному коду с использованием программы «Перевод тетракода в 2/10 форматы чисел» [20]: *а* – тетракод кодирует одно значение; *б* – тетракод кодирует совокупность значений



а)



б)

Рисунок 2.10 — Результаты приведения тетракода к двоичному коду с использованием программы «Перевод тетракода в 2/10 форматы чисел» [20]: тетракод, состоящий из разрядов неопределенности, каждый раз возвращает случайное значение

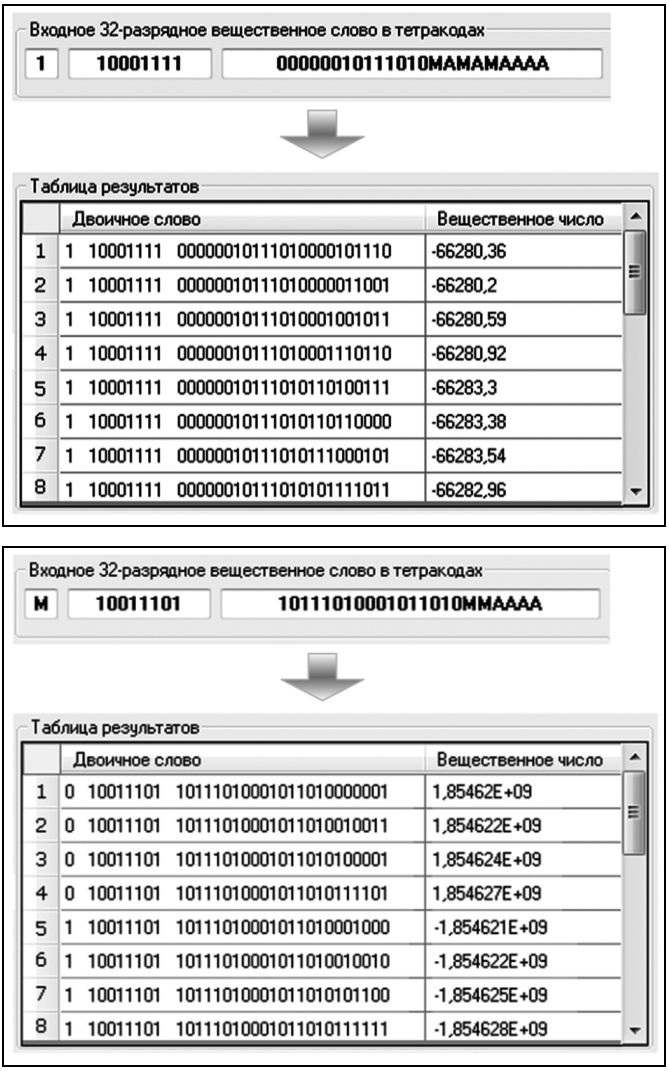


Рисунок 2.11 — Результаты приведения тетракода к формату чисел с плавающей запятой одинарной точности с использованием программы «Перевод тетракода в 2/10 форматы чисел» [20]

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

1) **Кодирование тетракода или постбинарное кодирование** — это процесс приведения одного или нескольких двоичных кодов к тетракоду.

2) **Декодирование тетракода** или **постбинарное декодирование** — это процесс приведения тетракода к одному или нескольким двоичным кодам.

3) **Представление тетракода** — это поразрядная замена тетракода одним или несколькими значениями другой кодовой системы (запись тетракода, т. е. представление его «внешнего вида» в другой кодовой системе).

2.4. Запись тетракодов с использованием двоичного и шестнадцатеричного кодирования

Реализация постбинарных вычислений, в частности возможность работы с тетракодами на современных электронно-вычислительных системах (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 [21, 22] использовалось следующее пары бит для кодирования одного тетрита: для тетрита А — 00; для тетрита М — 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 (рис. 2.12).

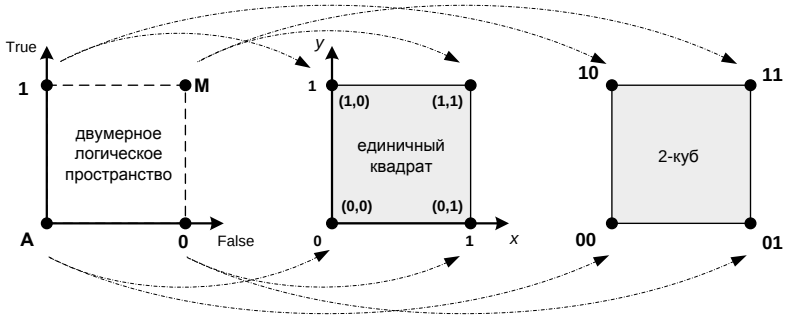


Рисунок 2.12 — Сопоставление состояний двумерного логического пространства координатам единичного квадрата и вершинам 2-куба

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 2.4. 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.

Таблица 2.4 — Таблица кодирования разрядов тетракода (Tetra) с использованием 16-ричных (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

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: 10 01 10 10 11 00 00 00.

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 т. ч. 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 кодах 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 $10101001t \neq 1001100110010110b$.

2.5. Принципы постбинарного кодирования десятичных чисел

Частым случаем ошибок представления, имеющих инженерный характер, являются ошибки перевода из одной системы счисления в другую. Многие десятичные дроби не имеют точного конечного представления среди двоичных чисел, которыми оперируют современные компьютерные системы. Но при введении подобных чисел в вычислительные системы 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 [23]. Например, величина a, представляющая неопределенность значений на числовой прямой между 1 и 5, может быть задана $1 \leq a \leq 5$ или, в более 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. 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.

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 [6].

В [3, 17] рассмотрен процесс перехода от десятичного интервала к тетракоду на примере вычисления иррационального числа π с помощью формулы Бэйли-Боруэйна-Плаффа [24].

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 значащими цифрами, полученными на первом и сотом шагах вычислений по формуле Бэйли-Боруэйна-Плаффа:

Двоичные отображения различающихся, начиная с одной десятитысячной, 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:

$x_1 = 3{,}1414225_{10} = 11.00100100001\mathbf{\underline{0}}10001000011110101\text{b}$,

$x_2 = 3{,}1415927_{10} = 11.00100100001\mathbf{\underline{1}}11101101011010011\text{b}$,

откуда

T = 11.001001000011MMMMMAAAAAAAAAAAAA.

Позиция первого (старшего) значения М определена первым несовпадением битовых значений в соответствующих разрядах (в двоичных значениях x_1 и x_2 эти разряды подчеркнуты).

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 (2.9).

$$\operatorname{mid}\boldsymbol{x} = \frac{1}{2}\left(x_1 + x_2\right), \qquad (2.9)$$

На рис. 2.13 показан процесс формирования тетракода Т с использованием двоичных значений x_1, x_2 и $\operatorname{mid}\boldsymbol{x}$. Позиция старшего М определяется равенством (2.10) или первым несовпадением значений разрядов одного веса:

$$m = \left\lfloor \lg 2^n \right\rfloor, \qquad (2.10)$$

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

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.

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.

Рисунок 2.13 — Кодирование интервала $\boldsymbol{x}$ тетракодом T с последующим получением границ $t$min, $t$max результирующего интервала

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 (рис. 2.13).

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 разделе, 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 свою 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 $f(n)$ по Бэйли-Боруэйна-Плаффу.

На основании полученных результатов можно утверждать, что при кодировании исходного десятичного интервала $\boldsymbol{x}$ тетракодом T и последующим приведением этого тетракода к результирующему десятичному интервалу $\boldsymbol{t}$ справедливо следующее соотношение:

$$\boldsymbol{t} \supseteq \boldsymbol{x}. \qquad (2.11)$$

Условие (2.11) лежит в основе постбинарного кодирования десятичных чисел. Соответственно, на основании вышесказанного, при формировании тетракода можно руководствоваться следующим:

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 (2.11) 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 (2.11).

На рис. 2.14 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 рис. 2.14 неравенств зависимость

$$\boldsymbol{t}' \supset \boldsymbol{t} \supset \boldsymbol{y} \qquad (2.12)$$

подтверждает приведенное выше высказывание под вторым пунктом и не противоречит условию (2.11).

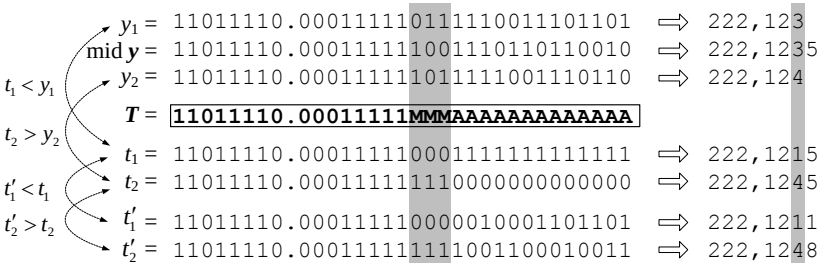


Рисунок 2.14 — Кодирование интервала y тетракодом T с последующим декодированием к интервалам $\boldsymbol{t}$ и $\boldsymbol{t}'$

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 электронно-вычислительными системами, удваивается по сравнению с двоичным кодом, повышение степени информативности получаемых за счет этого тетракодов вполне оправдывает увеличение затрат на кодирование.

2.6. Особенности приведения тетракодов к интервальным значениям (декодирование тетракодов)

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.

Такая 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 (рис. 2.15 *б*). Тетракод, состоящий только из значений 0 и 1 называется **вырожденным** и проводится соответственно к вырожденному интервалу, т. е. к одному значению (рис. 2.15 *в*). И только в случае соблюдения нормированности тетракода (рис. 2.15 *а*) справедливо его приведение к границам интервала.

а)

б)

в)

Рисунок 2.15 — Примеры нормированных (*а*), ненормированных (*б*) и вырожденных (*в*) тетракодов

Процесс приведения нормированного тетракода к значениям границ интервального числа (декодирование тетракода) является задачей, обратной к рассмотренной в п. 2.5 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 М, приведенных к двоичным единицам. При этом ширина wid полученного интервала $\boldsymbol{x}$ может быть вычислена по формуле (2.9), 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 (рис. 2.16 *а*). 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 М в тетракоде, а точнее — от позиции старшего тетрита поля М.

Поэтому

$$\mathrm{wid}\,\boldsymbol{x} = 2^{k+1} - 1, \qquad (2.13)$$

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

Нормированный тетракод, содержащий поля М и А 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 $\mathrm{wid}\,\boldsymbol{x}^{\min}$ и $\mathrm{wid}\,\boldsymbol{x}^{\max}$ соответственно (рис. 2.16 *б*):

$$q = x_1^{\max} - x_1^{\min} = x_2^{\max} - x_2^{\min}. \qquad (2.14)$$

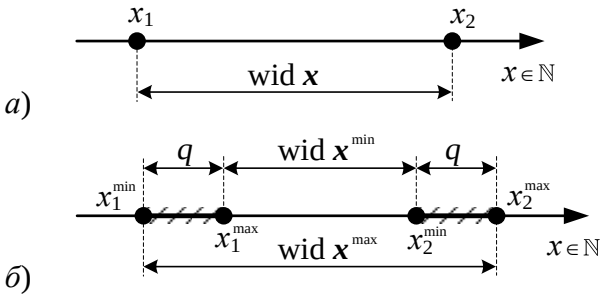


Рисунок 2.16 — Формирование целочисленных интервальных границ при декодировании нормированного тетракода содержащего только группу М (*а*) и группы М и А (*б*) [3]

При декодировании нормированного тетракода с полями М и А к интервальным границам, расстояние $\mathrm{wid}\,\boldsymbol{x}^{\max}$ между крайними точками $x_1^{\min}$ и $x_2^{\max}$ определяется аналогично (2.14), 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): $\mathrm{wid}\,\boldsymbol{x}^{\max} = \mathrm{wid}\,\boldsymbol{x}$.

Расстояние $\text{wid}\,\boldsymbol{x}^{\min}$ между точками $x_1^{\max}$ и $x_2^{\min}$ связано с $\text{wid}\,\boldsymbol{x}^{\max}$ соотношением

$$\text{wid}\,\boldsymbol{x}^{\min} = \text{wid}\,\boldsymbol{x}^{\max} - 2q, \qquad (2.15)$$

откуда

$$q = \frac{1}{2}\cdot(\text{wid}\,\boldsymbol{x}^{\max} - \text{wid}\,\boldsymbol{x}^{\min}). \qquad (2.16)$$

Равенство (2.16) определяет q как расстояние между двумя интервалами шириной $\text{wid}\,\boldsymbol{x}^{\min}$ и $\text{wid}\,\boldsymbol{x}^{\max}$. При этом, согласно свойствам интервального анализа, q — это расстояние на множестве $\mathbb{IR}$ как целочисленных, так и вещественных интервалов (поскольку $\mathbb{R} \supset \mathbb{Z}$, то $\mathbb{IR} \supset \mathbb{IZ}$).

Точки $x_1^{\min}$ и $x_2^{\max}$ сформированы с позиции учета погрешности, вносимого 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 $\text{wid}\,\boldsymbol{x}^{\max}$ и q, при которых представленные тетракодом границы интервала будут находиться в выделенных на рис. 2.15 *б* участках числовой оси.

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

$$q = 2^{l+1} - 1, \qquad (2.17)$$

где l — позиция старшего тетрита равного А в n-разрядном тетракоде: $k - 1 \le l \le 0$ (k — позиция старшего тетрита равного М).

При этом значение $\mathrm{wid}\,\boldsymbol{x}^{\min}$ можно определить, объединив соотношения (2.15) и (2.17):

$$\mathrm{wid}\,\boldsymbol{x}^{\min} = 2^{k+1} - 1 - 2 \cdot (2^{l+1} - 1) = 2^{k+1} - 2^{l+2} + 1. \qquad (2.18)$$

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

$$[\, x_i^{\min},\ x_i^{\max} \,] = [\, x_i^{\max} - q,\ x_i^{\max} \,] = [\, x_i^{\min},\ x_i^{\min} + q \,] \qquad (2.19)$$

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

В формулах (2.13), (2.17) и (2.18) используются переменные k и l, в которые подставляются номера позиций старших разрядов тетракода, равных М и А соответственно. Но иногда проще оперировать не позициями разрядов, а количеством тетритов в полях М и А.

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

$$\mathrm{wid}\,\boldsymbol{x} = 2^u - 1; \qquad (2.20)$$

$$q = 2^v - 1; \qquad (2.21)$$

$$\mathrm{wid}\,\boldsymbol{x}^{\max} = 2^{u+v} - 1; \qquad (2.22)$$

$$\operatorname{wid} \boldsymbol{x}^{\min} = 2^{u+v} - 2^{v+1} + 1. \qquad (2.23)$$

На рис. 2.17 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 $\boldsymbol{x}^{\max}$. 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 $w_2 = 17$.

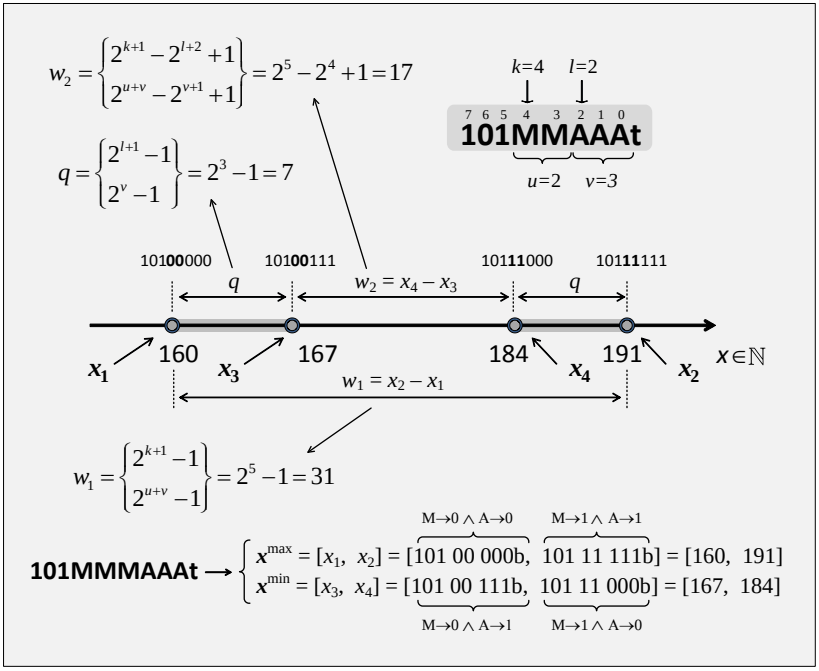


Рисунок 2.17 — Пример расчета предельных значений границ и диапазона их возможного изменения для интервала, декодированного из тетракода 101MMAAA

При этом диапазон возможного изменения границ интервала, декодированного из 101MMAAAt следующий: $160 \div 167$ для левой границы; $184 \div 191$ для правой границы. Согласно табл. 2.4, тетракод 101MMAAA в памяти вычислительного устройства 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 к границам интервала входящих в области нормализованных (имеющих ненулевое значение в поле порядка) и денормализованных (имеющих нулевое значение в поле порядка) чисел с плавающей запятой [25].

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 [3, 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:

$$\operatorname{wid} \boldsymbol{x}' = \xi_{subnorm,\, norm} \cdot \operatorname{wid} \boldsymbol{x}; \qquad (2.24)$$

$$\operatorname{wid} \boldsymbol{x}'^{\min} = \xi_{subnorm,\, norm} \cdot \operatorname{wid} \boldsymbol{x}^{\min}; \qquad (2.25)$$

$$\text{wid}\, \boldsymbol{x}'^{\max} = \xi_{subnorm,\, norm} \cdot \text{wid}\, \boldsymbol{x}^{\max}, \qquad (2.26)$$

где $\xi_{subnorm,\, norm}$ — коэффициент, учитывающий особенности формата представления чисел с плавающей запятой стандарта IEEE 754-2008 [25]: $\xi_{subnorm}$ — для денормализованных («субнормальных», *subnormal*) чисел; ξ_{norm} — для нормализованных (нормальных, *normal*) чисел.

Таким образом,

$$\xi_{subnorm} = 2^{1-offset-m}; \qquad (2.27)$$

$$\xi_{norm} = 2^{E_{10}-offset-m}, \qquad (2.28)$$

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 (2.16) значения $\text{wid}\, \boldsymbol{x}^{\min}$ и $\text{wid}\, \boldsymbol{x}^{\max}$ по (2.25) и (2.26):

$$q' = \frac{1}{2} \cdot (\text{wid}\, \boldsymbol{x}'^{\max} - \text{wid}\, \boldsymbol{x}'^{\min}) = \xi_{subnorm,\, norm} \cdot q. \qquad (2.29)$$

Следует отметить, что зависимость (2.24) применима для нормированного тетракода, содержащего только поле M, а зависимости (2.25) и (2.26) — 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.

На рис. 2.17 приведены все расчеты при декодировании 32-разрядного тетракода к интервалу, границы которого заданы в формате с плавающей запятой одинарной точности. 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: $0{,}21441650 \div 0{,}21442030$ для левой границы; $0{,}21447372 \div 0{,}21447752$ для правой границы.

В памяти бинарного компьютера приведенный на рис. 2.18 тетракод, согласно табл. 2.4, будет представлен следующей последовательностью бит:

$$01011010101010010110011010011010$$
$$10010110111111110000000000000000_2,$$

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 [3, с. 168–171] рассмотрен случай декодирования тетракода, содержащего в знаковом разряде значение M.

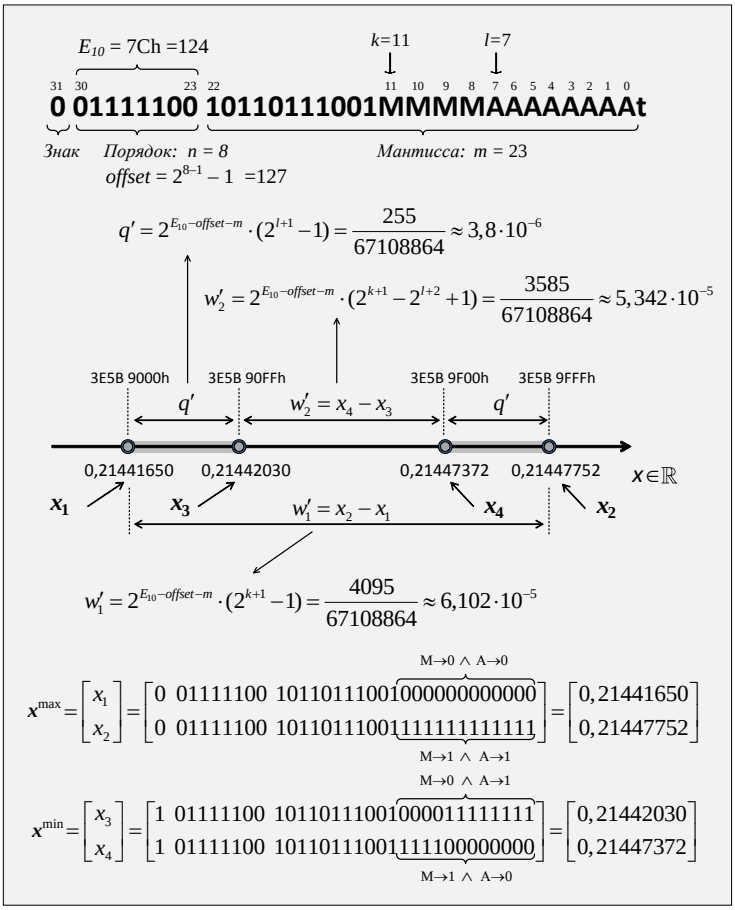


Рисунок 2.18 — Пример расчета предельных значений границ и диапазона их возможного изменения для вещественного интервала, декодированного из 32-разрядного тетракода

Такой тетракод отвечает обозначенным выше требованиям к нормированности (относящимся, прежде всего, к полю мантиссы). При формировании значения левой границы тетрит M 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 рис. 2.19 показаны получения значений границ «крайне широкого» (рис. 2.18 *а*), «крайне узкого» (рис. 2.19 *б*) и одного из множества возможных (поле A заполнено случайными двоичными значениями) результирующих интервалов (рис. 2.19 *в*) при декодировании «вещественного» тетракода. В последующем, при выполнении арифметических операций над тетракодами анализу будет подвергаться обязательное сохранение числовых значений, принадлежащих «крайне узкому» 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 [3, с. 171–172] 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. представлять двоичное значение одной из интервальных границ.

Рисунок 2.19 — Результаты приведения тетракода к вещественным интервалам, границы которых заданы в формате числа с плавающей запятой одинарной точности с использованием программы «Перевод тетракода в 2/10 форматы чисел» [20]

Таким образом, используя комплекс соотношений (2.17) – (2.29), можно провести интервальное оценивание тетракода и получить необходимые метрики результирующего интервального числа, не прибегая при декодировании к его полной «распаковке», заключающейся в получении двоичных или десятичных значений интервальных границ. При этом эффективное хранение интервального числа в тетракоде позволяет оперировать интервальными объектами в постбинарных компьютерных системах.

2.7. Определение и свойства арифметических операций над тетракодами

Поскольку тетракод $C_4 = \{0, A, M, 1\}$ используется для хранения числовой информации, над ним могут быть определены арифметические операции.

Пусть множество $o\rho = \{+, –, \times, \div\}$ 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 $o\rho$ справедливо большинство свойств аналогичных арифметических операций, определенных над действительными числами. Однако, ввиду «интервальной природы» тетракода (результатом декодирования тетракода в большинстве случаев является

интервал), операции $\circ\rho$ также должны соответствовать свойствам определенных над интервалами арифметических операций (рис. 2.20):

$$\text{wid}\,(\boldsymbol{x}_I \pm \boldsymbol{x}_J) = \text{wid}\,\boldsymbol{x}_I + \text{wid}\,\boldsymbol{x}_J; \quad (2.30)$$

$$\text{wid}\,(\boldsymbol{x}_I \times \boldsymbol{x}_J) \geq \max\{|\boldsymbol{x}_I| \times \text{wid}\,\boldsymbol{x}_I,\ |\boldsymbol{x}_J| \times \text{wid}\,\boldsymbol{x}_J\}; \quad (2.31)$$

$$\text{wid}\,(\boldsymbol{x}_I \div \boldsymbol{x}_J) \leq \{|\boldsymbol{x}_I| \times \text{wid}\,\boldsymbol{x}_I,\ |\boldsymbol{x}_J| \times \text{wid}\,\boldsymbol{x}_J\}. \quad (2.32)$$

Порядок выполнения операций $\circ\rho$ 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 $\circ\rho$ обязательно выполнение свойств коммутативности

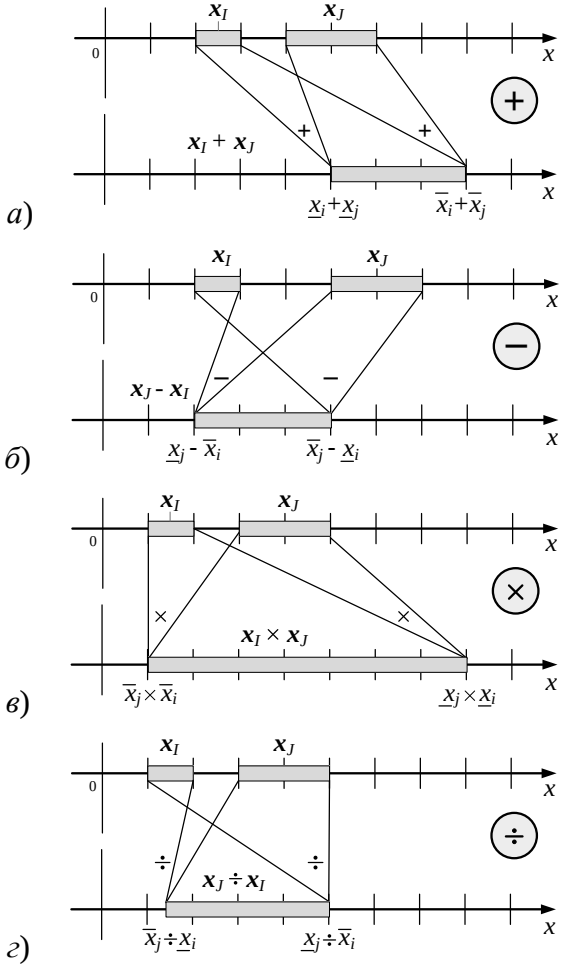
$$T_1 + T_2 = T_2 + T_1; \quad (2.33)$$

$$T_1 \times T_2 = T_2 \times T_1, \quad (2.34)$$

и ассоциативности

$$\left(T_1 + T_2\right) + T_3 = T_1 + \left(T_2 + T_3\right) = T_1 + T_2 + T_3; \qquad (2.35)$$

$$\left(T_1 \times T_2\right) \times T_3 = T_1 \times \left(T_2 \times T_3\right) = T_1 \times T_2 \times T_3. \qquad (2.36)$$



а)

б)

в)

г)

Рисунок 2.20 — Графическая интерпретация интервальных арифметических операций (*а* – сложение, *б* – вычитание, *в* – умножение, *г* – деление)

Аналогично интервальной арифметике, для операций $\circ\rho$ свойство дистрибутивности имеет место в ослабленном виде [26]:

$$T_1 \times (T_2 + T_3) \subset T_1 \times T_2 + T_1 \times T_3. \qquad (2.37)$$

Выражение (2.37) указывает о субдистрибутивности умножения относительно сложения.

2.8. Арифметическое сложение тетракодов

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

На рис. 2.21 представлены два интервала, полученные при приведении тетракода T к двоичному коду.

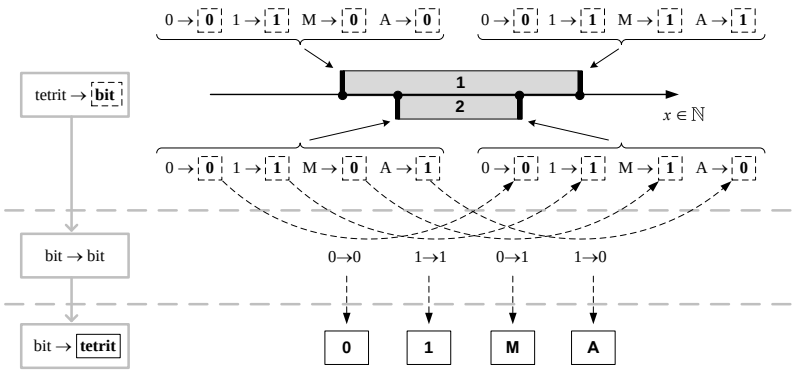


Рисунок 2.21 — Битовое (bit) представление тетритов (tetrit) в тетракоде по отношению к границам интервалов с максимальной (интервал 1) и минимальной (интервал 2) шириной

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 $1+1=10$.

Поскольку интервал 2 является «крайне узким» (рис. 2.15), то все его значения будут принадлежать всем возможным интервалам, полученным при декодировании тетракода T. Поэтому, если $\boldsymbol{t}_1^{\min}$ является «крайне узким» интервалом, декодированным из тетракода T_1, а $\boldsymbol{t}_2^{\min}$ — «крайне узкий» интервал из T_2, то результатом сложения $T_1 + T_2$ является такой тетракод T_3, при декодировании которого полученный «крайне узкий» интервал $\boldsymbol{t}_3$ представляет собой сумму интервалов $\boldsymbol{t}_1^{\min}$ и $\boldsymbol{t}_2^{\min}$. Следует отметить, что при любых значениях двух тетракодов всегда найдется третий тетракод, являющийся суммой двух предыдущих:

$$\forall\ T_1 \rightarrow \boldsymbol{t}_1^{\min},\ T_2 \rightarrow \boldsymbol{t}_2^{\min}\ \exists\ T_3 \rightarrow \boldsymbol{t}_3^{\min}:$$
$$T_1 + T_2 = T_3 \wedge\ \boldsymbol{t}_1^{\min} + \boldsymbol{t}_2^{\min} = \boldsymbol{t}_3^{\min}. \qquad (2.38)$$

Запись (2.38) 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 рис. 2.21 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 рис. 2.20 для интервала № 2 и формирование интервалов, имеющих двоичные значения границ.

Шаг 2. Выполнение операции сложения над сформированными интервалами по формуле $[x_1, x_2]+[x_3, x_4]= =[x_1+x_3, x_2+x_4]$ 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 рис. 2.21):

- 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 M или A:

  $(\textit{бит}\ 0 \to \textit{бит}\ 1) \to \textit{тетрит}\ \text{M},$

  $(\textit{бит}\ 1 \to \textit{бит}\ 0) \to \textit{тетрит}\ \text{A}.$

В левой части рис. 2.21 фактически показано поэтапное выполнение приведенного выше алгоритма сложения.

На рис. 2.22 показано получение суммы для некоторых пар тетритов [27, с. 97–104], а в табл. 2.5 сведены результаты сложений тетритов во всех возможных сочетаниях. Из рис. 2.22 видно, что сложение для тетритов коммутативно, поскольку коммутативно двоичное сложение.

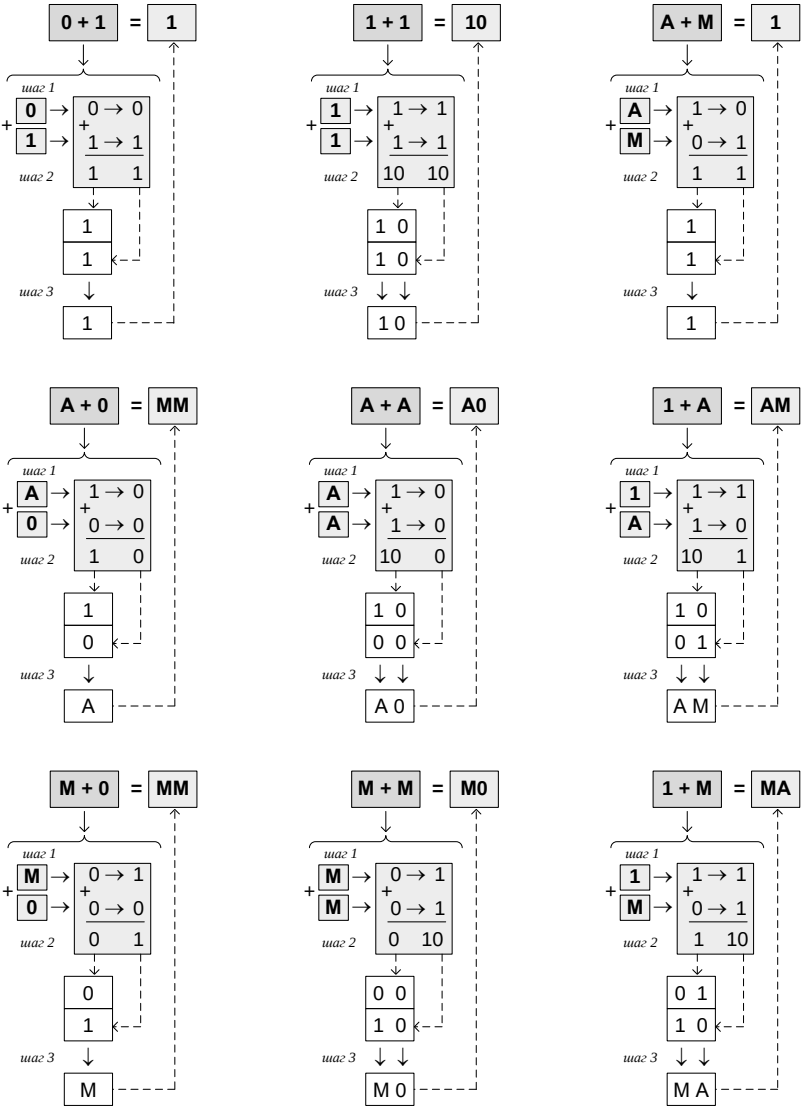


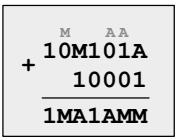
Рисунок 2.22 — Схематическая реализация алгоритма выполнения операции сложения двух тетритов

Сложение тетракодов выполняется поразрядно, начиная с младшего разряда и используя значения табл. 2.5. На рис. 2.23 и 2.24 приведены два примера сложения тетракодов с демонстрацией проверки правильности полученной суммы путем сложения «крайне узких» интервалов и сопоставления полученных результатов.

Таблица 2.5 — Таблица сложения тетритов: $t_1 + t_2$

t_2 \ t_1	0	A	M	1
0	0	A	M	1
A	A	A0	1	AM
M	M	1	M0	MA
1	1	AM	MA	10

Пример 1



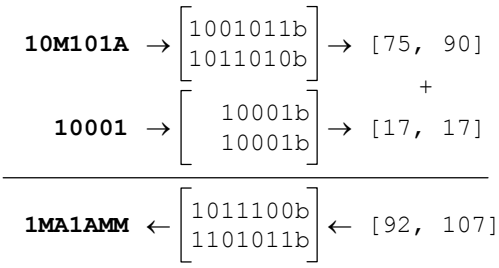


Рисунок 2.23 — Пример 1: сложение тетракодов с демонстрацией проверки правильности полученной суммы

Пример 2

```
   М М
  1010МММ
+   100ММА
  --------
  111ММ01
```

$$\mathbf{1010MMM} \rightarrow \begin{bmatrix} 1010000b \\ 1010111b \end{bmatrix} \rightarrow [80,\ 87]$$

$$+$$

$$\mathbf{100MMA} \rightarrow \begin{bmatrix} 100001b \\ 100110b \end{bmatrix} \rightarrow [33,\ 38]$$

$$\mathbf{111MM01} \leftarrow \begin{bmatrix} 1110001b \\ 1111101b \end{bmatrix} \leftarrow [113,\ 125]$$

Рисунок 2.24 — Пример 2: сложение тетракодов с демонстрацией проверки правильности полученной суммы

1+1+A =	(1+1)+A = 10+A=**1A** 1+(A+1) = 1+AM=**1A**
1+1+M =	(1+1)+M = 10+M=**1M** 1+(1+M) = 1+MA=**1M**
A+M+1 =	(A+M)+1 = 1+1=**10** A+(M+1) = A+MA=**10**
A+M+M =	(A+M)+M = 1+M=**MA** A+(M+M) = A+M0=**MA**
A+A+M =	(A+A)+M = A0+M=**AM** A+(A+M) = A+1=**AM**

Рисунок 2.25 — Примеры, показывающие выполнение свойства ассоциативности сложения

Поскольку сложение интервальных чисел обладает свойствами ассоциативности, сложение тетракодов также не противоречит свойству (2.37). Это подтверждается

примерами на рис. 2.25. В табл. 2.6 показана связность полученных результатов сложения тетритов из табл. 2.5 с результатами операции суммы по модулю 2 тетралогики из табл. 2.3.

Таблица 2.6 — 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

$\oplus$	0	A	M	1
0	0	A	M	1
A	A	0	1	M
M	M	1	0	A
1	1	M	A	0

2.9. 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 (табл. 2.5). Результат приведен в табл. 2.6 (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 2.7 — Таблица вычитания тетритов

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

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

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

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

Эти утверждения делают возможной проверку вычитания тетракодов путем вычитания по формуле $[x_1, x_2] - [x_3, x_4] = [x_1 - x_4, x_2 - x_3]$ «крайне узких» интервалов, декодированных из тетракодов-операндов. Аналогично записи (2.38) для сложения, при вычитании тетракодов справедлива следующая запись:

$$\forall\ T_1 \to \boldsymbol{t}_1^{\min},\ T_2 \to \boldsymbol{t}_2^{\min}:\ \boldsymbol{t}_1^{\min} \geq \boldsymbol{t}_2^{\min}\ \exists\ T_3 \to \boldsymbol{t}_3:$$

$$T_1 - T_2 = T_3 \wedge \boldsymbol{t}_1^{\min} - \boldsymbol{t}_2^{\min} \supseteq \boldsymbol{t}_3. \qquad (2.39)$$

Выражение (2.39) показывает, что если интервал $\boldsymbol{t}_1^{\min}$ является «крайне узким» интервалом, декодированным из тетракода T_1, а $\boldsymbol{t}_2^{\min}$ — «крайне узким» 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 «крайне узкий» 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.

Вычитание тетракодов выполняется поразрядно, начиная с младшего разряда, используя при этом значения табл. 2.7. На рис. 2.26 приведены два примера вычитания тетракодов с проверкой правильности полученных результатов.

В архитектуре постбинарной вычислительной системы, также как и в архитектуре современной (бинарной) ЭВМ **реализация вычитающих схем является избыточной**. Поэтому для упрощения архитектуры 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:

$$B_1 - B_2 = B_1 + (-B_2) = B_1 + (B_2)_{\text{ДК}}, \qquad (2.40)$$

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

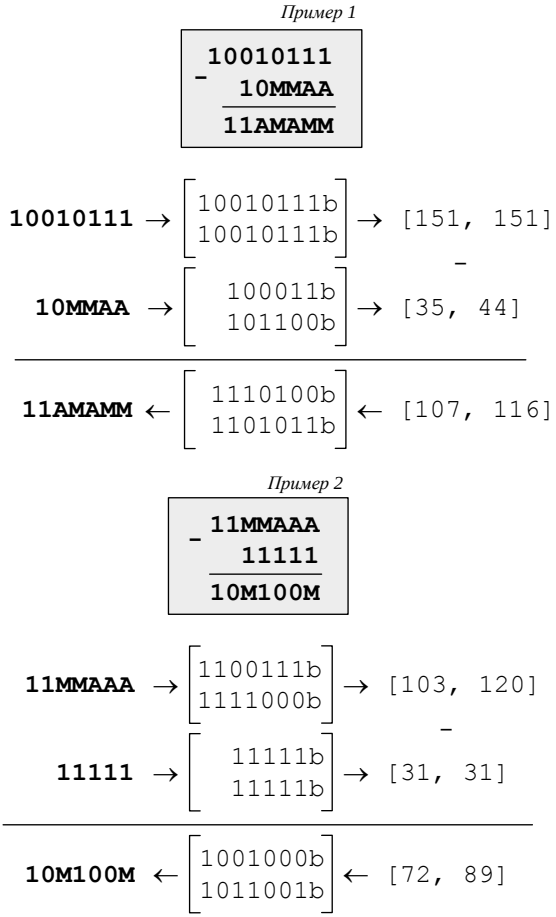


Рисунок 2.26 — Два примера вычитания тетракодов с демонстрацией проверки правильности полученной разности

Поскольку сложение и вычитание тетритов производится аналогично сложению и вычитанию битов, то для тетракодов T_1 и T_2 равенство (2.40) можно записать следующим образом:

$$T_1 - T_2 = T_1 + (-T_2) = T_1 + (T_2)_{\text{ДК}}, \qquad (2.41)$$

где запись $(T_2)_{\text{ДК}}$ подразумевает значение тетракода T_2 в дополнительном коде.

Таким образом, **дополнительный код тетракода** имеет тот же смысл, что и в двоичном коде (т. е. дополнение до 2, англ. *two's complement* [28]), 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* [28]) 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 из рис. 2.26, вычитание

$$10010111\text{t} - 10\text{MMAAt} =$$

$$= 0.10010111\text{t} - 0.0010\text{MMAAt} = 0.011\text{AMAMMt}$$

можно заменить операцией сложения в дополнительном коде

$$0.10010111\text{t} + (1.1101\text{AAMMt} + 1\text{t}) =$$

$$= 0.10010111\text{t} + 1.1101\text{A10At} = 0.011\text{AMAMMt}.$$

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

В табл. 2.8 приведены результаты умножения двух тетритов, а в табл. 2.9 — идентичность полученных результатов с результатами операции умножения тетралогики из табл. 2.1.

Таблица 2.8 — Таблица умножения тетритов: $t_1 \times t_2$

t_2 \ t_1	0	A	M	1
0	0	0	0	0
A	0	A	0	A
M	0	0	M	M
1	0	A	M	1

Умножение тетракодов, также как и сложение, эквивалентно интервальному умножению, о чем свидетельствует запись (2.42). Она показывает, что если интервал $\boldsymbol{t}_1$ является «крайне узким» интервалом, декодированным из тетракода T_1, а $\boldsymbol{t}_2$ — «крайне узким» интервалом из T_2, то результатом умножения $T_1 \times T_2$ является такой тетракод T_3, при декодировании которого

полученный «крайне узкий» интервал $\boldsymbol{t}_3$ представляет собой произведение интервалов $\boldsymbol{t}_1^{\min} \cdot \boldsymbol{t}_2^{\min}$. Произведение интервалов выполняется по формуле

$$[x_1, x_2] \times [x_3, x_4] = [\min(x_1x_3, x_1x_4, x_2x_3, x_2x_4), \max(x_1x_3, x_1x_4, x_2x_3, x_2x_4)].$$

Следует отметить, что при любых значениях двух тетракодов всегда найдется третий тетракод, являющийся произведением двух предыдущих:

$$\forall\ T_1 \to \boldsymbol{t}_1^{\min},\ T_2 \to \boldsymbol{t}_2^{\min}\ \ \exists\ T_3 \to \boldsymbol{t}_3^{\min}:$$
$$T_1 \times T_2 = T_3 \wedge \boldsymbol{t}_1^{\min} \cdot \boldsymbol{t}_2^{\min} = \boldsymbol{t}_3^{\min}. \qquad (2.42)$$

Таблица 2.9 — 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

На рис. 2.27 показаны два примера умножения тетракодов с демонстрацией проверки правильности полученного произведения через умножение интервальных значений с последующим постбинарным кодированием интервала-результата. Поскольку умножение интервальных чисел обладает свойством ассоциативности, умножение тетракодов также не противоречит свойству (2.36), что подтверждается примерами на рис. 2.28.

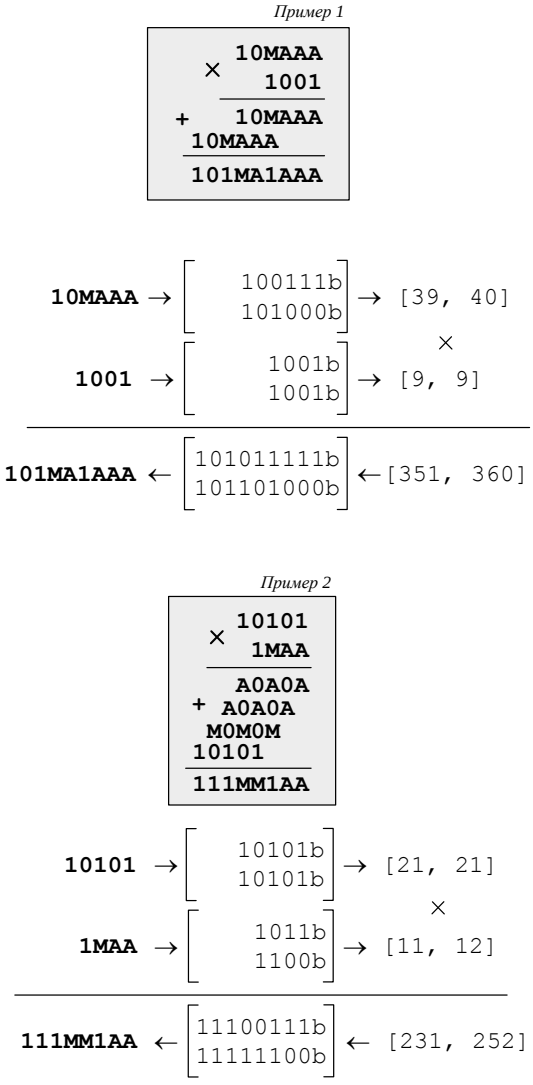


Рисунок 2.27 — Примеры умножения тетракодов с демонстрацией проверки правильности полученного произведения

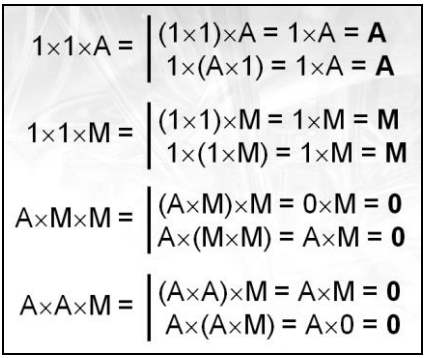


Рисунок 2.28 — Примеры, показывающие выполнение свойства ассоциативности умножения

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. 2.7 связь между умножением и делением, тетракоды из примеров рис. 2.27 можно связать операцией деления (рис. 2.29, 2.30).

При делении тетракодов справедлива следующая запись (2.43)

$$\forall\ T_1 \to \boldsymbol{t}_1^{\min},\ T_2 \to \boldsymbol{t}_2^{\min}:\ \exists\ T_3 \to \boldsymbol{t}_3:$$
$$T_1 \div T_2 = T_3 \wedge \boldsymbol{t}_1^{\min} \div \boldsymbol{t}_2^{\min} \supseteq \boldsymbol{t}_3, \qquad (2.43)$$

которая показывает, что если интервал $\boldsymbol{t}_1^{\min}$ является «крайне узким» интервалом, декодированным из тетракода T_1, а $\boldsymbol{t}_2^{\min}$ — «крайне узким» 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 «крайне узкий» интервал $\boldsymbol{t}_3$ представляет собой подинтервал интервала, полученного при делении $\boldsymbol{t}_1^{\min}$ на $\boldsymbol{t}_2^{\min}$.

Пример 1

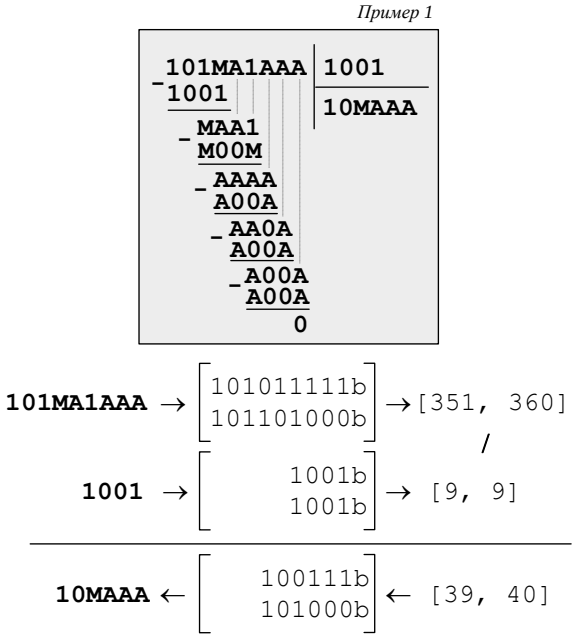


Рисунок 2.29 — Пример 1: деление тетракодов с демонстрацией проверки правильности полученного частного

Запись $\boldsymbol{t}_1^{\min} \div \boldsymbol{t}_2^{\min} \supseteq \boldsymbol{t}_3$ 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.

Пример 2

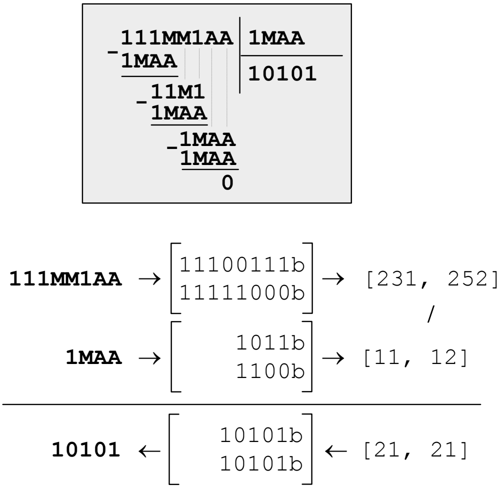


Рисунок 2.30 — Пример 2: деление тетракодов
с демонстрацией проверки правильности
полученного частного

2.11. Выводы

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 тетрафукциями (иметь два аргумента), так и тернарными (иметь три аргумента) или *n*-арными (иметь *n* аргументов).
4. Унарные инверсные и сдвиговые операции тетралогики, а также 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.

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

Полученные в пунктах 2.1–2.6 результаты кодирования и декодирования тетракодов, а также приведенные в п. 2.6 зависимости и соотношения позволяют сформировать следующие заключения:

1. Кодирование числовой информации с помощью четырехзначной кодовой системы {0, A, M, 1} эффективно только при использовании основных принципов тетракдирования.
2. Применительно к тетракодам справедливы следующие действия: кодирование, декодирование и представление.
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 «крайне широкого» и «крайне узкого» 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.

Глава 3
ПРОГРАММНО-АППАРАТНАЯ РЕАЛИЗАЦИЯ

Если вы будете работать для настоящего,
то ваша работа выйдет ничтожной;
надо работать, имея в виду
только будущее

Антон Чехов
(из дневниковых записей)

3.1. Разработка и проектирование базовых элементов тетралогики

Концепция тералогики впервые была сформулирована в 1996 году в работе [1] и в последующем существенно эволюционировала. С самого начала в качестве э**лементов тетралогики** рассматривались устройства, выполняющие функции (операции) тетралогики над входными сигналами (операндами, данными) и предназначенные для обработки информации в цифровой постбинарной форме.

В рамках данной работы цифровой сигнал для элементов тетралогики, как сигнал данных, у которого каждый из представляющих параметров описывается функцией дискретного времени и конечным множеством возможных значений, будет задаваться последовательностью сигналов высокого — «В» и низкого

— «Н» уровней цифрового сигнала, что соответствует значениям единицы и нуля двоичной логики. Последнее обосновано тем, что элементы тетралогики, как составляющие постбинарного компьютинга, реализуются на базе современных двоичных компьютерных систем. При этом состояние тетралогики, аналогично определенному порядку в табл. 2.4, кодируются парой двоичных разрядов [2, с. 260–273]: «00» — для неопределенности A; «01» — для тетрануля; «10» — для тетраединицы; «11» — для множественности M.

Проектирование компьютерных компонентов для реализации логических операций тетралогики включает в себя следующие этапы:

1. Формирование классической таблицы истинности для тетрафункции с учетом замены каждого тетрита парой двоичных значений.
2. Формирование аналитического выражения, являющегося основой для синтеза с использованием канонической нормальной формы представления логических функций [3].
3. Синтез комбинационной схемы и ее проектирование в среде разработки, моделирования и верификации проектов для программируемых логических интегральных схем Active-HDL компании Aldec, с использованием языка описания аппаратуры интегральных схем VHDL [4].

В табл. 3.1 представлена таблица истинности для унарных тетрафункций: инверсия (pNOT), максимизация и минимизация неопределенности (MAX_A, MIN_A) и множественности (MAX_M, MIN_M) [5, с. 136], а в табл. 3.2 — для бинарных тетрафункций: конъюнкция

(pAND), дизъюнкция (pOR) и исключающее ИЛИ (сумма по модулю 2, pXOR) [2, с. 256–257, с. 270–272].

Таблица 3.1 — Таблица истинности для унарных тетрафункций (в скобках приведены значения тетритов)

Входные значения	*Выходные значения*				
	минимизации		*максимизации*		*инверсии*
	$q = MIN\_A(a)$	$r = MIN\_M(a)$	$s = MAX\_A(a)$	$v = MAX\_M(a)$	$w = pNOT(a)$
a[1:0]	q[1:0]	r[1:0]	s[1:0]	v[1:0]	w[1:0]
00 (A)	01 (0)	00 (A)	10 (1)	00 (A)	11 (M)
01 (0)	01 (0)	01 (0)	01 (0)	01 (0)	10 (1)
10 (1)	10 (1)	10 (1)	10 (1)	10 (1)	01 (0)
11 (M)	11 (M)	01 (0)	11 (M)	10 (1)	00 (A)

Из табл. 3.1 получаем следующие аналитические выражения (все операции в данных выражениях — операции булевой алгебры):

$$q_0 = a_0 \vee \overline{a}_1 = \overline{\overline{a}_0 \wedge a_1}; \quad (3.1)$$

$$q_1 = a_1; \quad (3.2)$$

$$r_0 = a_0; \quad (3.3)$$

$$r_1 = \overline{a}_0 \wedge a_1 = \overline{a_0 \vee \overline{a}_1}; \quad (3.4)$$

$$s_0 = a_0; \tag{3.5}$$

$$s_1 = \overline{a}_0 \vee a_1 = \overline{a_0 \wedge \overline{a}_1}; \tag{3.6}$$

$$v_0 = a_0 \wedge \overline{a}_1 = \overline{\overline{a}_0 \vee a_1}; \tag{3.7}$$

$$v_1 = a_1; \tag{3.8}$$

$$w_0 = \overline{a}_0; \tag{3.9}$$

$$w_1 = \overline{a}_1. \tag{3.10}$$

На основании выражений (3.1–3.10) получены комбинационные схемы, которые приведены на рис. 3.1. На рис. 3.2 приведены созданные в Active-HDL компоненты для унарных функций тетралогики с демонстрацией результатов моделирования.

Из табл. 3.2 получаем следующие аналитические выражения (все операции также являются операциями булевой алгебры):

$$x_0 = a_0 \vee b_0; \tag{3.11}$$

$$x_1 = a_1 \wedge b_1; \tag{3.12}$$

$$y_0 = a_0 \wedge b_0; \tag{3.13}$$

$$y_1 = a_1 \vee b_1; \tag{3.14}$$

$$z_0 = \left(\overline{a}_0 \wedge \overline{b}_0\right) \vee \left(a_0 \wedge b_0\right); \tag{3.15}$$

$$z_1 = \left(\overline{a}_1 \wedge b_1\right) \vee \left(a_1 \wedge \overline{b}_1\right). \tag{3.16}$$

На основании аналитических выражений (3.11–3.14) для бинарных тетрафункций pAND и pOR спроектированы комбинационные схемы, которые приведены на рис. 3.3. На рис. 3.4 приведены созданные в Active-HDL компоненты для конъюнкции и дизъюнкции тетралогики с демонстрацией результатов моделирования.

Реализация функции pXOR не нуждается в синтезе отдельного логического элемента, так как может быть

реализована с использованием уже разработанных компонентов pAND, pOR и pNOT, поскольку

$$\begin{aligned} &\mathrm{pXOR}(a, b) = \\ &= \mathrm{pOR}(\mathrm{pAND}(\mathrm{pNOT}(a), b), \mathrm{pAND}(a, \mathrm{pNOT}(b))). \end{aligned} \tag{3.17}$$

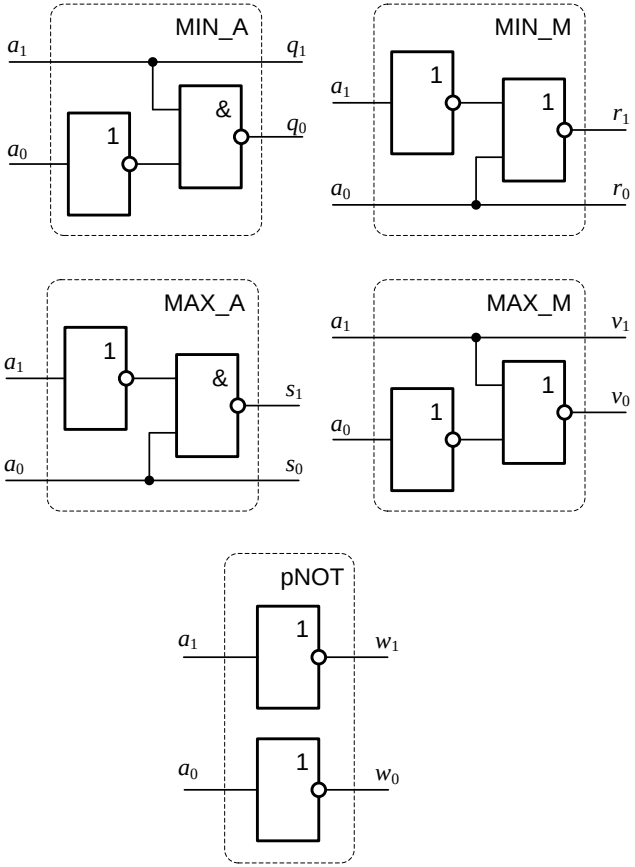


Рисунок 3.1 — Комбинационные схемы логических элементов, реализующих унарные операции тетралогики

Справедливость выражения (3.17) может быть доказана подстановкой полученных ранее зависимостей (3.9–3.14) для функций pAND, pOR и pNOT.

Таблица 3.2 — Таблица истинности для бинарных тетрафункций (в скобках приведены значения тетритов)

Входные значения		*Выходные значения*		
		постбинарной конъюнкции: x = = pAND(a, b)	*постбинарной дизъюнкции: y = = pOR(a, b)*	*постбинарной суммы по модулю 2 z = = pXOR(a, b)*
a[1:0]	*b*[1:0]	*x*[1:0]	*y*[1:0]	*z*[1:0]
00 (A)	00 (A)	00 (A)	00 (A)	01 (0)
00 (A)	01 (0)	01 (0)	00 (A)	00 (A)
00 (A)	10 (1)	00 (A)	10 (1)	11 (M)
00 (A)	11 (M)	01 (0)	10 (1)	10 (1)
01 (0)	00 (A)	01 (0)	00 (A)	00 (A)
01 (0)	01 (0)	01 (0)	01 (0)	01 (0)
01 (0)	10 (1)	01 (0)	10 (1)	10 (1)
01 (0)	11 (M)	01 (0)	11 (M)	11 (M)
10 (1)	00 (A)	00 (A)	10 (1)	11 (M)
10 (1)	01 (0)	01 (0)	10 (1)	10 (1)
10 (1)	10 (1)	10 (1)	10 (1)	01 (0)
10 (1)	11 (M)	11 (M)	10 (1)	00 (A)
11 (M)	00 (A)	01 (0)	10 (1)	10 (1)
11 (M)	01 (0)	01 (0)	11 (M)	11 (M)
11 (M)	10 (1)	11 (M)	10 (1)	00 (A)
11 (M)	11 (M)	11 (M)	11 (M)	01 (0)

Так, для пар значений (a_0, b_0) и (a_1, b_1) получаем

$$\begin{aligned}
&\mathrm{pXOR}(a_0, b_0) = \\
&= \mathrm{pOR}(\mathrm{pAND}(\overline{a}_0, b_0), \mathrm{pAND}(a_0, \overline{b}_0)) = \\
&= \mathrm{pOR}(\overline{a}_0 \vee b_0, a_0 \vee \overline{b}_0) = \left(\overline{a}_0 \vee b_0\right) \wedge \left(a_0 \vee \overline{b}_0\right) = \\
&= \left(\overline{a}_0 \wedge \overline{b}_0\right) \vee \left(a_0 \wedge b_0\right).
\end{aligned} \tag{3.18}$$

$$\begin{aligned}
&\mathrm{pXOR}(a_1, b_1) = \\
&= \mathrm{pOR}(\mathrm{pAND}(\overline{a}_1, b_1), \mathrm{pAND}(a_1, \overline{b}_1)) = \\
&= \mathrm{pOR}(\overline{a}_1 \wedge b_1, a_1 \wedge \overline{b}_1) = \left(\overline{a}_1 \wedge b_1\right) \vee \left(a_1 \wedge \overline{b}_1\right).
\end{aligned} \tag{3.19}$$

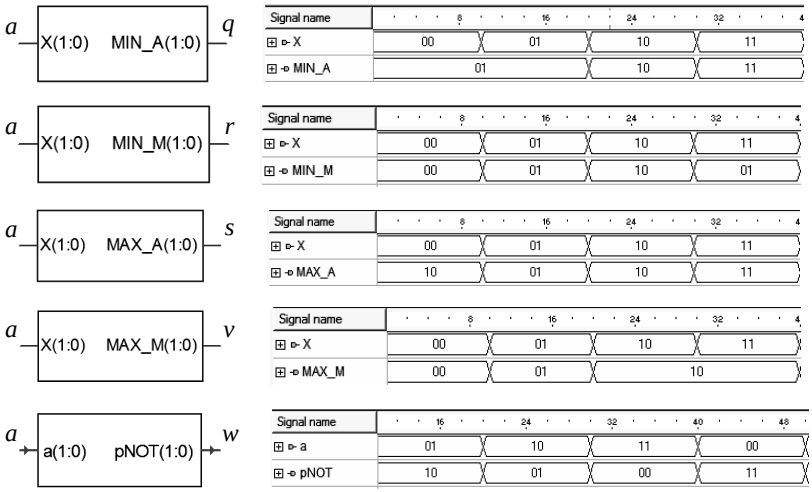


Рисунок 3.2 — Созданные в Active-HDL компоненты (слева), реализующие унарные функции тетралогики и диаграммы моделирования (справа)

Сопоставляя результаты выражений (3.18), (3.19) и (3.15), (3.16), очевидно, что pXOR (a_0, b_0) = z_0 и

pXOR (a_1, b_1) = z_1. Учитывая, что z = pXOR (a, b) (табл. 3.2), справедливость выражения (3.17) доказана.

На рис. 3.5 в Active-HDL выполнена практическая реализация зависимости (3.17) с демонстрацией результатов моделирования.

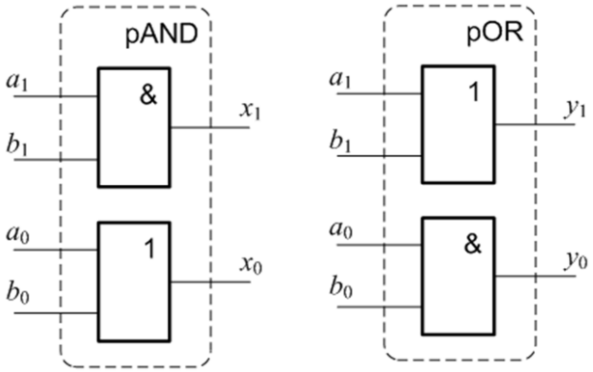


Рисунок 3.3 — Комбинационные схемы логических элементов, реализующих конъюнкцию и дизъюнкцию тетралогики

Таблица 3.3 представляет собой таблицу истинности для унарных тетрафункций инверсной группы [5, с. 136–137]: симметричных (SWAP_A/M, SWAP_0/1) и вероятностных (SWAP_FL, SWAP_TR).

Следующие аналитические выражения получены из табл. 3.3 с применением в некоторых выражениях операции отрицания булевой алгебры:

$$k_0 = a_1; \tag{3.20}$$

$$k_1 = a_0; \tag{3.21}$$

$$l_0 = \overline{a}_1; \tag{3.22}$$

$$l_1 = \overline{a}_0; \tag{3.23}$$

$$m_0 = \overline{a}_0; \tag{3.24}$$

$$m_1 = a_1; \tag{3.25}$$

$$n_0 = a_0; \tag{3.26}$$

$$n_1 = \overline{a}_1. \tag{3.27}$$

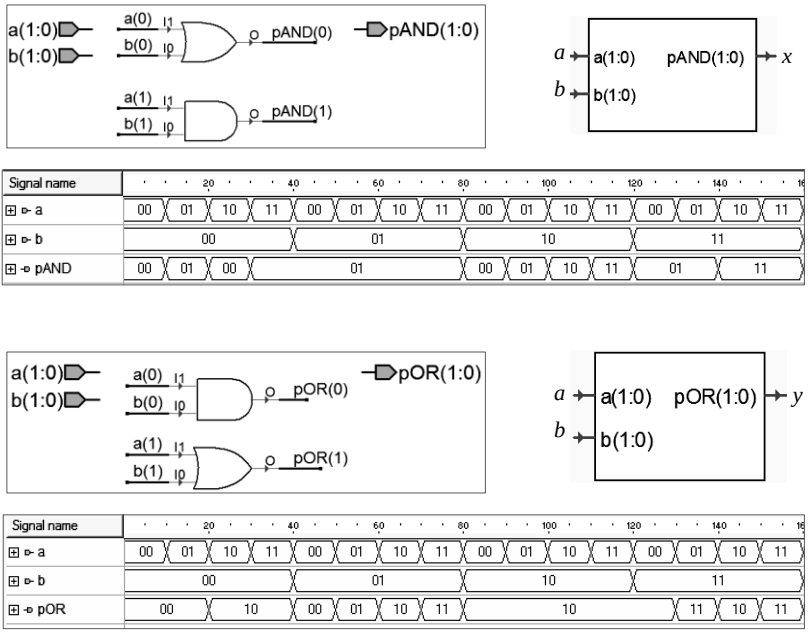


Рисунок 3.4 — Созданные в Active-HDL компоненты, реализующие унарные функции тетралогики: конъюнкцию (вверху) и дизъюнкцию (внизу)

Комбинационные схемы, построенные на основании выражений (3.20–3.27), приведены на рис. 3.6.

Для унарных сдвиговых логических операций (см. п. 2.3) определены направления «B» (сдвиг «назад», совпадающий с направлением оси «*False*» двумерного логического пространства) и «F» (сдвиг «вперед», совпадающий с направлением оси «*True*» двумерного логического пространства) [5, 6].

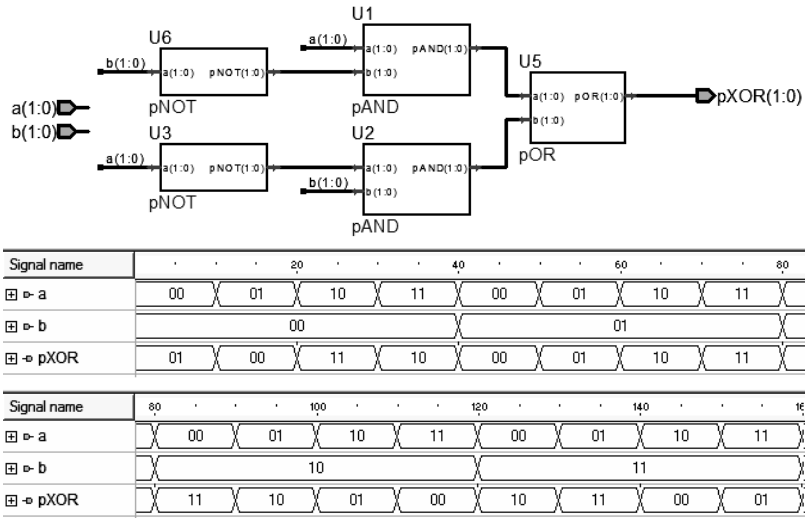


Рисунок 3.5 — Реализация операции тетралогики pXOR с использованием компонентов pAND, pOR и pNOT

Так, логическая операция SHIFT_<*i*>B(*a*) определяет сдвиг тетрита *a* «назад» на *i* разрядов, а операция SHIFT_<*i*>F(*a*) — сдвиг *a* «вперед» на *i* разрядов ($i \in \mathbb{N}$). В табл. 3.4 приведены результаты выполнения операций SHIFT_<*i*>F(*a*) и SHIFT_<*i*>B(*a*) для всех возможных входных значениях *a* и при количествах сдвигов, равных *i* mod 4.

Таблица 3.3 — Таблица истинности для инверсных унарных тетрафункций (в скобках приведены значения тетритов)

Входные значения	*Выходные значения*			
	симметричной инверсии		*вероятностной инверсии*	
	k = *SWAP_A/M(a)*	l = *SWAP_0/1(a)*	m = *SWAP_FL(a)*	n = *SWAP_TR(a)*
a[1:0]	k[1:0]	l[1:0]	m[1:0]	n[1:0]
00 (A)	00 (A)	11 (M)	01 (0)	10 (1)
01 (0)	10 (1)	01 (0)	00 (A)	11 (M)
10 (1)	01 (0)	10 (1)	11 (M)	00 (A)
11 (M)	11 (M)	00 (A)	10 (1)	01 (0)

Сопоставляя данные таблиц 3.1, 3.3 и 3.4, можно убедиться в справедливости равенств (3.28) и (3.29).

$$\mathrm{SHIFT\_}\langle i\rangle\mathrm{F}(a)=\begin{cases} a, \ if \ \ i \bmod 4 = 0, \\ \mathrm{SWAP\_0/1(SWAP\_FL}(a)), \ if \ \ i \bmod 4 = 1, \\ \mathrm{pNOT}(a), \ if \ \ i \bmod 4 = 2, \\ \mathrm{SWAP\_0/1(SWAP\_TR}(a)), \ if \ \ i \bmod 4 = 3. \end{cases} \tag{3.28}$$

$$\mathrm{SHIFT\_}\langle i\rangle\mathrm{B}(a)=\begin{cases} a, \ if \ \ i \bmod 4 = 0, \\ \mathrm{SWAP\_0/1(SWAP\_TR}(a)), \ if \ \ i \bmod 4 = 1, \\ \mathrm{pNOT}(a), \ if \ \ i \bmod 4 = 2, \\ \mathrm{SWAP\_0/1(SWAP\_FL}(a)), \ if \ \ i \bmod 4 = 3. \end{cases} \tag{3.29}$$

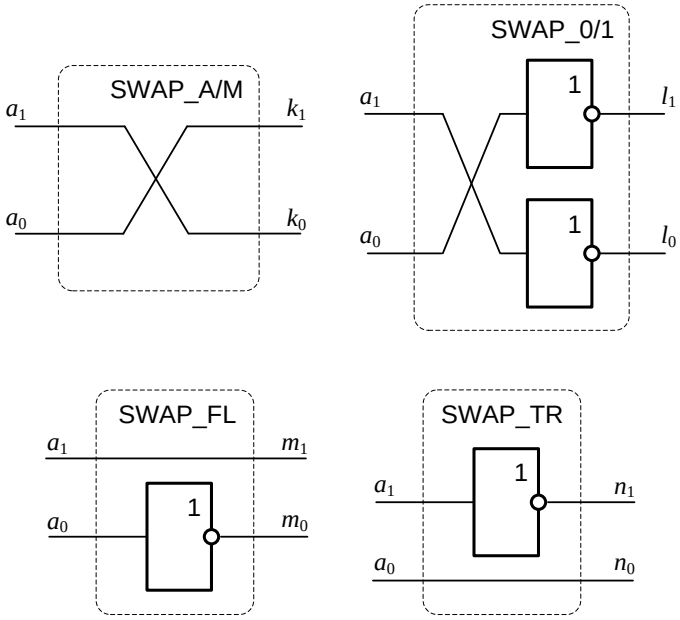


Рисунок 3.6 — Комбинационные схемы логических элементов, реализующих операции тетралогики инверсной группы

Из (3.28) и (3.29) следует, что для аппаратной реализации операций SHIFT_<i>F(a) и SHIFT_<i>B(a) нужно использовать компоненты SWAP_0/1, SWAP_FL, SWAP_TR и pNOT. Также можно сделать заключение, что тождественное равенство

$$SHIFT\_\langle i\rangle \mathrm{F}(a) \equiv \mathrm{SHIFT}\_\langle j\rangle \mathrm{B}(a) \qquad (3.30)$$

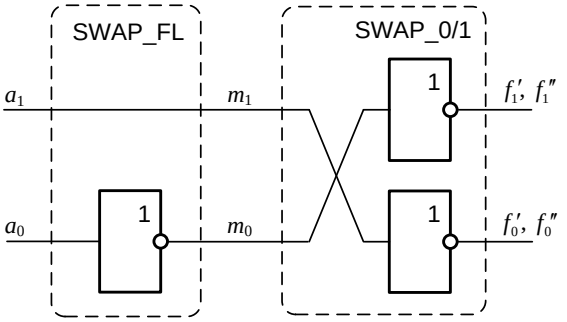
справедливо, тогда и только тогда, когда значения i и j связаны соотношением

$$j = \begin{cases} 0, \ if \ \ i \bmod 4 = 0, \\ 4 - i \bmod 4, \ if \ \ i \bmod 4 \neq 0. \end{cases} \qquad (3.31)$$

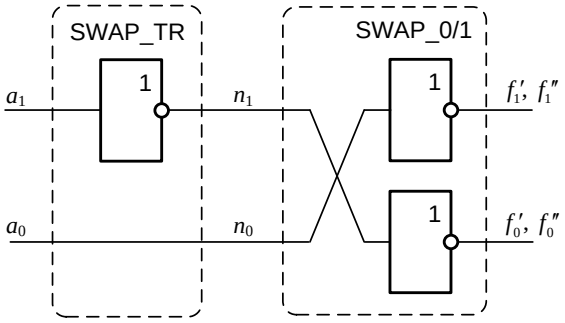
Таблица 3.4 — Таблица истинности для сдвиговых унарных тетрафункций (в скобках приведены значения тетритов)

Входные значения	*Сдвиг*	*Выходные значения*	
		f' = *SHIFT_<i>F(a)*	f'' = *SHIFT_<i>B(a)*
a[1:0]	*i* mod 4	f' [1:0]	f'' [1:0]
00 (A)	0	00 (A)	00 (A)
01 (0)		01 (0)	01 (0)
10 (1)		10 (1)	10 (1)
11 (M)		11 (M)	11 (M)
00 (A)	1	01 (0)	10 (1)
01 (0)		11 (M)	00 (A)
10 (1)		00 (A)	11 (M)
11 (M)		10 (1)	01 (0)
00 (A)	2	11 (M)	11 (M)
01 (0)		10 (1)	10 (1)
10 (1)		01 (0)	01 (0)
11 (M)		00 (A)	00 (A)
00 (A)	3	10 (1)	01 (0)
01 (0)		00 (A)	11 (M)
10 (1)		11 (M)	00 (A)
11 (M)		01 (0)	10 (1)

Рисунок 3.7 иллюстрирует равенства (3.30) и (3.31) для нетривиальных композиций функции при *i* mod 4 = {1, 3} с учетом тождества (3.30).



$$f' = SHIFT\_1F(a)$$
$$f'' = SHIFT\_3B(a)$$



$$f' = SHIFT\_3F(a)$$
$$f'' = SHIFT\_1B(a)$$

Рисунок 3.7 — Комбинационные схемы, реализующие операции тетралогики сдвиговой группы

Очевидно, что схемы, приведенные на рис. 3.7, можно упростить так, чтобы использовать только логические элементы SWAP_FL и SWAP_TR (рис. 3.8).

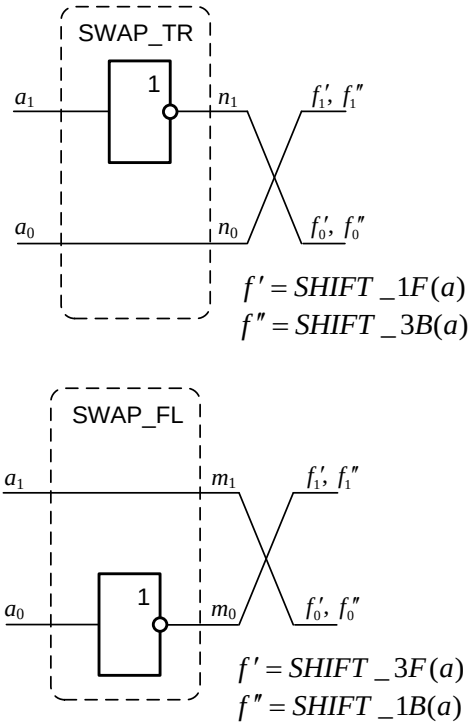


Рисунок 3.8 — Результат упрощения комбинационных схем, реализующих операции тетралогики сдвиговой группы

3.2. Синтез постбинарных суммирующих компонентов

Операции алгебраического сложения занимают основное место среди всех операций, выполняемых современными бинарными компьютерными системами.

В постбинарном компьютинге в качестве способа представления данных выступает тетракод (п. 2.4),

представленный комбинацией тетритов. При этом тетрит, как один разряд тетракода, является единицей измерения количества информации и может принимать значения тетрануля (0), тетраединицы (1), неопределенности (А) и множественности (М) [7]. Поэтому реализация операции сложения тетракодов также имеет фундаментальное значение и является основой тетравычислений [2], поскольку операции вычитания, умножения и деления тетракодов, в конечном счете, сводятся к сложению.

Возможность реализации постбинарных суммирующих компонентов основывается на следующих предпосылках:

1. Поскольку в основе операции сложения находится операция суммирования одноразрядных чисел, то в работе [8] был предложен принцип нахождения суммы тетритов с доказательством соблюдения свойств сложения в арифметике.
2. В ряде работ [5, 9, 10] на основе математической логики и аксиоматики теории множеств были сформулированы аспекты алгебры тетралогики и синтезированы блоки, реализующие основные тетралогические операции с использованием базовых логических элементов И, ИЛИ, НЕ.
3. Первые попытки реализации постбинарных суммирующих элементов были выполнены в работе [2, с. 248–258], однако не были получены законы их функционирования для создания функциональных схем, и, соответственно, не были выполнены оценки аппаратных затрат и расчет временных параметров.

Реализацию схем постбинарных цифровых компонентов целесообразно выполнять в полнофункциональном булевом базисе. В качестве такого

базиса выбрана функция И-НЕ (функция Шеффера) исходя из следующих соображений:

- вследствие функциональной полноты базиса И-НЕ реализующие их вентили могут представлять любую булеву операцию И, ИЛИ, НЕ и таким образом самостоятельно образовать базис, в котором реализуется любая логическая функция;
- при проектировании логических схем можно обойтись одним единственным типом вентиля, что позволяет предельно унифицировать этот процесс;
- для большинства серий ТТЛ- и КМОП-логик вентиль И-НЕ является базисным и предпочтителен во многих отношениях, вследствие чего реализация логических схем в базисе И-НЕ получила широкое распространение в практике [11, с. 42–44].

Таким образом, для получения суммирующих постбинарных компонентов необходимо выполнить следующие действия:

1. Получить уравнения, описывающие закон функционирования постбинарных одноразрядных (т. е. однотетритных) суммирующих компонентов: постбинарных полусумматора и сумматора.
2. Выполнить реализацию полученных логических функций в полнофункциональном базисе И-НЕ и построить схемы постбинарных суммирующих компонентов, состоящих из вентилей И-НЕ.
3. Произвести оценку аппаратурных затрат и временных задержек полученных схем и выполнить

сравнительный анализ с эквивалентными двоичными суммирующими схемами.

Аналогично двоичному полусумматору, **постбинарный полусумматор** представляет собой простейшее цифровое устройство, используемое для организации постбинарного сумматора.

Постбинарный одноразрядный (однотетритный) полусумматор принимает на вход два тетрита и формирует результат их сложения (сумма *S*) и возникающий при сложении перенос (*C*). Таблица истинности постбинарного полусумматора приведена в табл. 3.5 [2, с. 102; 9].

Таблица 3.5 — Таблица истинности постбинарного полусумматора

X	*Y*	*S*	*C*
A	A	0	A
A	0	A	0
A	1	M	A
A	M	1	0
0	A	A	0
0	0	0	0
0	1	1	0
0	M	M	0

X	*Y*	*S*	*C*
1	A	M	A
1	0	1	0
1	1	0	1
1	M	A	M
M	A	1	0
M	0	M	0
M	1	A	M
M	M	0	M

Для перехода к булевым функциям, задающим закон функционирования постбинарного полусумматора, применяется система кодирования одного тетрита парой двоичных разрядов [2, с. 249]. Таким образом, тетрит *K* постбинарного компьютинга может быть заменен парой бит *k*[1:0] (табл. 3.6) для дальнейшей реализации

постбинарных цифровых устройств в базисе И-НЕ. Данная таблица полностью соотносима с табл. 2.4.

Таблица 3.6 — Таблица кодирования
тетрита K битами k[1:0]

K	k_1	k_0	K	k_1	k_0
A	0	0	1	1	0
0	0	1	M	1	1

На основании табл. 3.6, таблица истинности постбинарного полусумматора (табл. 3.5) может быть записана так, что в каждой ячейке окажутся только значения двоичных разрядов (табл. 3.7).

Из табл. 3.7 путем минимизации по картам Карно получены дизъюнктивные нормальные формы (ДНФ) булевых функций, задающих закон функционирования постбинарного полусумматора:

$$s_1 = \overline{x}_1 y_1 \vee x_1 \overline{y}_1; \tag{3.32}$$

$$s_0 = \overline{x}_0 \overline{y}_0 \vee x_0 y_0; \tag{3.33}$$

$$c_1 = x_1 y_1; \tag{3.34}$$

$$c_0 = x_0 \vee y_0. \tag{3.35}$$

Уравнение (3.32) определяет сумму по модулю 2, следовательно уравнение (3.32) представимо как $s_1 = x_1 \oplus y_1$. Аналогичным образом через сумму по модулю 2 можно представить и уравнение (3.33):

$$s_0 = \overline{\overline{\overline{x}_0 \overline{y}_0 \vee x_0 y_0}} = \overline{(x_0 \vee y_0) \cdot (\overline{x}_0 \vee \overline{y}_0)} =$$
$$= \overline{x_0 \overline{x}_0 \vee x_0 \overline{y}_0 \vee \overline{x}_0 y_0 \vee y_0 \overline{y}_0} =$$
$$= \overline{x_0 \overline{y}_0 \vee \overline{x}_0 y_0} = \overline{x_0 \oplus y_0}.$$

Таблица 3.7 — Таблица истинности постбинарного полусумматора с двоичными состояниями

x_1	x_0	y_1	y_0	s_1	s_0	c_1	c_0
0	0	0	0	0	1	0	0
0	0	0	1	0	0	0	1
0	0	1	0	1	1	0	0
0	0	1	1	1	0	0	1
0	1	0	0	0	0	0	1
0	1	0	1	0	1	0	1
0	1	1	0	1	0	0	1
0	1	1	1	1	1	0	1

x_1	x_0	y_1	y_0	s_1	s_0	c_1	c_0
1	0	0	0	1	1	0	0
1	0	0	1	1	0	0	1
1	0	1	0	0	1	1	0
1	0	1	1	0	0	1	1
1	1	0	0	1	0	0	1
1	1	0	1	1	1	0	1
1	1	1	0	0	0	1	1
1	1	1	1	0	1	1	1

Для реализации в функционально полном логическом базисе И-НЕ уравнения (3.32–3.35) должны быть преобразованы с использованием законов Де Моргана и двойной инверсии. Применение этих законов приводит к новой системе уравнений:

$$s_1 = \overline{\overline{\overline{x}_1 y_1} \cdot \overline{x_1 \overline{y}_1}}; \tag{3.36}$$

$$s_0 = \overline{\overline{\overline{x}_0 \overline{y}_0} \cdot \overline{x_0 y_0}}; \tag{3.37}$$

$$c_1 = \overline{\overline{x_1 y_1}}; \tag{3.38}$$

$$c_0 = \overline{\overline{x}_0 \overline{y}_0}. \tag{3.39}$$

Уравнения (3.36) и (3.38) (равно как и уравнения (3.32) и (3.34) задают закон функционирования классического двоичного полусумматора [12, с. 88], что позволяет использовать двоичный полусумматор при реализации постбинарного.

Функциональная схема постбинарного однотетритного полусумматора, основанная на уравнениях (3.36–3.39)

приведена на рис. 3.9 *а*. При этом верхняя часть схемы со входами x_1, y_1 и выходами s_1, c_1 является классической схемой двоичного полусумматора.

На рис. 3.9 *б* показано условное обозначение постбинарного полусумматора PHA (от англ. *Postbinary Half-Adder*). На рис. 3.9 *в* показаны условные обозначения классического двоичного полусумматора HA, формирующего старшие разряды суммы s_1 и переноса c_1, и постбинарной «надстройки» PFHA (от англ. *Postbinary Fraction Half-Adder*), которая формирует младшие разряды суммы s_0 и переноса c_0. Совместное использование элементов HA и PFHA открывает возможность реализации постбинарного полусумматора с использованием двоичного.

Для полусумматора PHA время t формирования суммы S равно

$$t_S = \max(t_{s_0}, t_{s_1}) = 3t_G, \tag{3.40}$$

а время t формирования переноса C равно

$$t_C = \max(t_{c_0}, t_{c_1}) = 2t_G, \tag{3.41}$$

где t_G — время задержки элемента И-НЕ.

Уравнения (3.40) и (3.41) позволяют сделать вывод, что быстродействие PHA равно быстродействию HA и равно трём временам задержки элемента И-НЕ: $\max(t_S, t_C) = 3t_G$.

Схема PHA имеет 12 логических элементов И-НЕ и цену по Квайну, равную 24, что примерно в 1,6 раза больше суммарного числа входов схемы HA [12, с. 90]. На основании значения цены по Квайну можно сделать вывод, что площадь, занимаемая схемой PHA незначительно превышает площадь, занимаемую схемой HA.

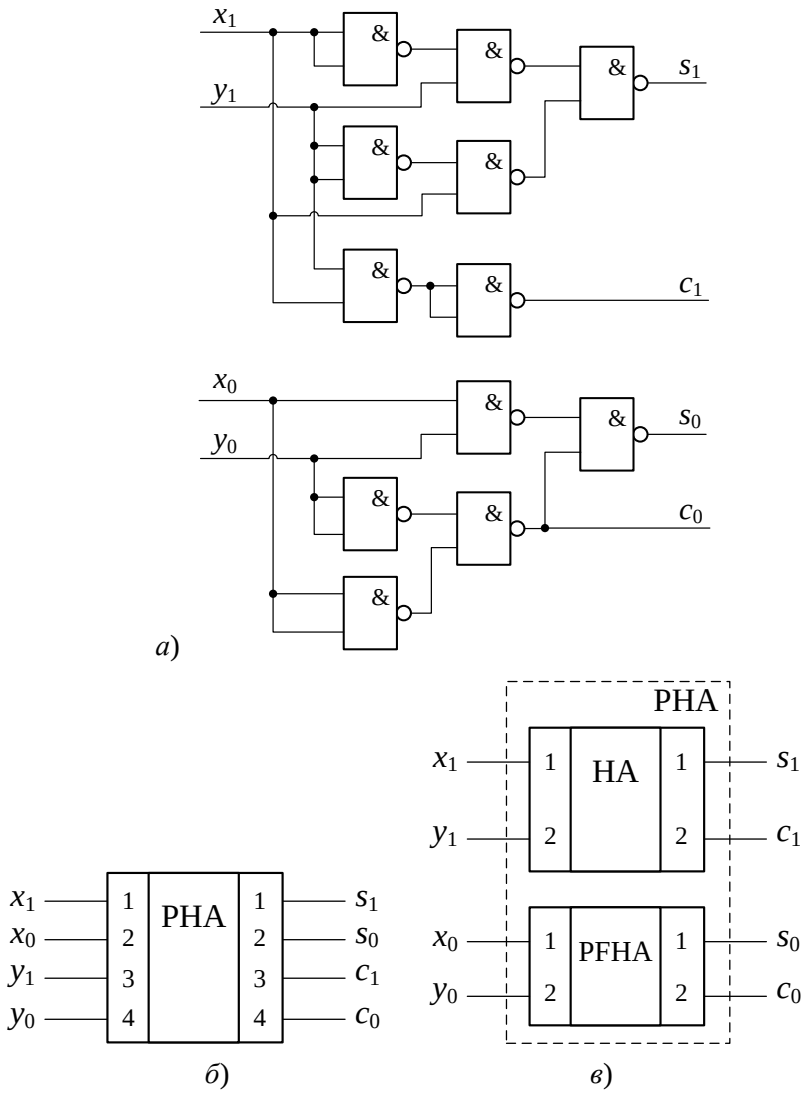


Рисунок 3.9 — Функциональная схема (*а*), условное графическое обозначение (*б*) постбинарного полусумматора и его замещение (*в*) двоичным полусумматором НА (Half-Adder) с добавочным элементом PFHA (Postbinary Fraction Half-Adder)

Поскольку значения x_0 и x_1 задают тетрит X, а y_0 и y_1 — тетрит Y (табл. 3.5, 3.7), то исходя из назначения постбинарного полусумматора, его условное обозначение на рис. 1*б* можно обобщить для входных тетритов X и Y с получением тетритов суммы S и переноса C (рис. 3.10).

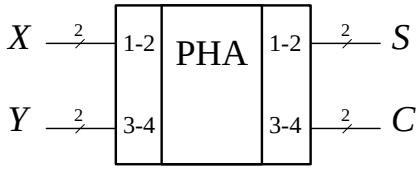


Рисунок 3.10 — Обобщенное условное обозначение постбинарного полусумматора

На рис. 3.11 представлены результаты моделирования постбинарного полусумматора PHA в среде Active-HDL.

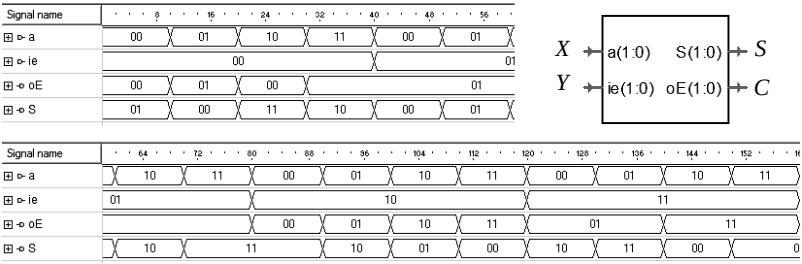


Рисунок 3.11 — Компонент PHA и диаграмма его работы

При сложении тетракодов $X_{n-1}X_{n-2}\ldots X_1X_0$ и $Y_{n-1}Y_{n-2}\ldots Y_1Y_0$ сумма каждого разряда тетракода определяется уравнением:

$$S_i = X_i + Y_i + C_{i-1},\ (i = \overline{0,n-1})\,. \qquad (3.42)$$

Перенос на текущем шаге $C_i = \{\text{A, M, 1}\}$ возникает при выполнении условия $S_i > 1$ ($i = \overline{0, n-1}$). Очевидно, что C_{i-1} — входной перенос (при $i = 0$) или перенос с предыдущего разряда.

Для определения значений суммы (3.42) и переноса при сложении одноразрядных тетракодов (тетритов) возможно использование **однотетритного постбинарного сумматора** (**тетрасумматора**).

При кодировании тетракодов двоичными числами (табл. 3.6), уравнение (3.42) можно представить как систему

$$\begin{cases} s_j = x_j + y_j + c_{j-2}; \\ s_{j+1} = x_{j+1} + y_{j+1} + c_{j-1}; \end{cases} \quad (j = 2i,\ i = \overline{0, n-1}). \qquad (3.43)$$

Система уравнений (3.43) позволяет задать таблицу истинности постбинарного сумматора. Любая таблица истинности описывает 2^p состояния, где p — мощность множества всех входных значений. Для системы (3.43) множество входов $\{x_{j+1}, x_j, y_{j+1}, y_j, c_{j-2}, c_{j-1}\}$ имеет мощность $p = 6$, поэтому таблица истинности одноразрядного постбинарного сумматора содержит $2^6 = 64$ состояния. При этом полученные из таблицы истинности канонические уравнения в ДНФ содержат по 32 терма для уравнений суммы и переноса постбинарного сумматора и являются бесполезными для дальнейшего синтеза. Минимизация ДНФ также не будет эффективной, поскольку окажется затруднительным оптимально использовать выбранный элементный базис для реализации схемы постбинарного сумматора.

Поэтому реализацию тетрасумматора можно осуществить, используя два постбинарных полусумматора РНА (рис. 3.10). Исходя из уравнения (3.42), первый РНА

определяет промежуточную сумму $S' = X_i + Y_i$, а второй РНА — окончательную сумму $S_i = S'_i + C_{i-1}$. При этом возможен перенос как от первого РНА (C_i^1), так и от второго РНА (C_i^2). Выходной перенос тетрасумматора C_i получается путем логического сложения тетритов C_i^1 и C_i^2. Такая операция в тетралогике называется постбинарной дизъюнкцией pOR [2, с. 256–258], следовательно C_i = pOR(C_i^1, C_i^2). Таким образом, справедлива схема, приведенная на рис. 3.12.

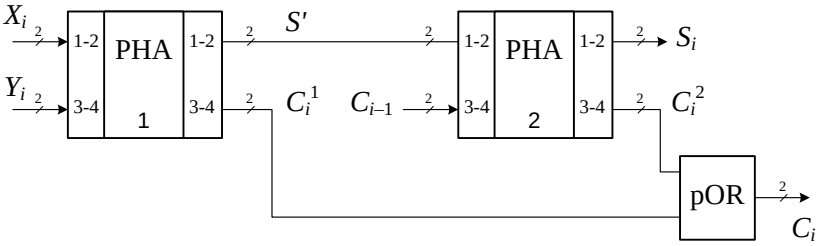


Рисунок 3.12 — Реализация тетрасумматора
на базе двух постбинарных полусумматоров РНА

На рис. 3.13 показана диаграмма работы компонента PFA (от англ. *Postbinary Full Adder*) из схемы на рис. 3.12, а временная диаграмма его работы полностью соответствует таблице истинности постбинарного сумматора.

При переходе к двоичному кодированию, уравнение (3.42) замещается системой уравнений (3.43), а выражение C_i = pOR(C_i^1, C_i^2), согласно [2, с. 256] сводится к системе булевых функций $(j = 2i,\ i = \overline{0, n-1})$:

$$\begin{cases} c_j = c_j^1 \cdot c_j^2; \\ c_{j+1} = c_{j+1}^1 \vee c_{j+1}^2. \end{cases} \qquad (3.44)$$

Учитывая системы уравнений (3.42) и (3.44), а также условное обозначение постбинарного полусумматора из рис. 3.9 *б*, реализацию тетрасумматора на базе двух постбинарных полусумматоров (рис. 3.12) можно представить в виде схемы на рис. 3.14.

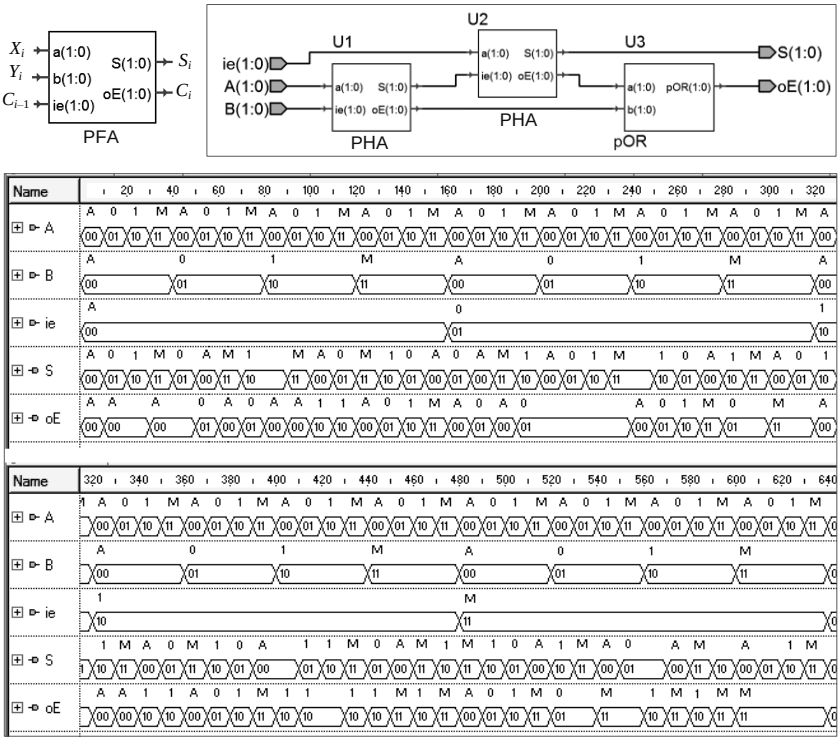


Рисунок 3.13 — Компонент PFA и диаграмма его работы

Если в полученной схеме каждый элемент PHA (рис. 3.9 *в*) разделить на элементы HA и PFHA, то можно выделить две подсхемы (рис. 3.15).

Верхняя подсхема является классической схемой реализации одноразрядного двоичного сумматора FA на базе двух полусумматоров HA [12, с. 91]. Нижняя схема PFFA (от англ. Postbinary Fraction Full Adder) является постбинарной «надстройкой» к сумматору FA. Исходя из уравнений (3.32) и (3.34) запишем уравнения, задающие закон функционирования двоичного сумматора FA (рис. 3.15):

$$s_{j+1} = s' \oplus c; . \tag{3.45}$$

$$c_{j+1} = c^1 \vee c^2 = xy \vee s'c; . \tag{3.46}$$

В уравнениях для удобства опущены индексы в правой части. Приведем (3.45) и (3.46) к следующему виду:

$$s_{j+1} = (x \oplus y) \oplus c = x \oplus y \oplus c; \tag{3.47}$$

$$\begin{aligned} c_{j+1} &= xy \vee (x \oplus y) \cdot c = xy \vee \bar{x}yc \vee x\bar{y}c = \\ &= (x \vee \bar{x}c) \cdot y \vee x\bar{y}c = xy \vee yc \vee x\bar{y}c = \\ &= (y \vee \bar{y}c) \cdot x \vee yc = xy \vee xc \vee yc. \end{aligned} \tag{3.48}$$

В уравнении (3.47) использован закон Блейка — Порецкого [11, с. 201]: $a \vee \bar{a}b = a \vee b$. Для подсхемы PFFA (рис. 3.15), согласно (3.33) и (3.35) получаем

$$s_j = \overline{s' \oplus c}; \tag{3.49}$$

$$c_j = c^1 \cdot c^2 = (x \vee y) \cdot (s' \vee c) \tag{3.50}$$

Ранее было отмечено, что $\overline{a \oplus b} = \bar{a}\bar{b} \vee ab$. Так, (3.49) и (3.50) можно привести к следующему виду:

$$s_j = \overline{\overline{x \oplus y} \oplus c} = \overline{\overline{x \oplus y} \cdot c \vee \overline{x \oplus y} \cdot \overline{c}} = \\ = \overline{(x \oplus y) \cdot c \vee \overline{x \oplus y} \cdot \overline{c}} = \overline{\overline{x \oplus y \oplus c}} = x \oplus y \oplus c; \quad (3.51)$$

$$c_j = (x \vee y) \cdot (\overline{x \oplus y} \vee c) = (x \vee y) \cdot (\overline{x}\,\overline{y} \vee xy \vee c) = \\ = x\overline{x}\,\overline{y} \vee xxy \vee xc \vee y\overline{x}\,\overline{y} \vee yxy \vee yc = xy \vee xc \vee yc. \quad (3.52)$$

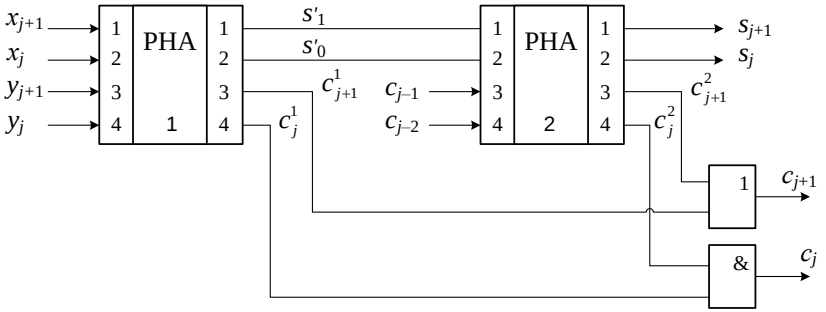


Рисунок 3.14 — Реализация тетрасумматора на базе двух постбинарных полусумматоров РНА с двоичными входами/выходами

Полученные уравнения равнозначны уравнениям (3.47) и (3.48) соответственно, поэтому они также задают закон функционирования двоичного сумматора. Значит подсхема PFFA (рис. 3.15) может быть замещена схемой сумматора FA. Восстанавливая индексы в правой части уравнений (3.47), (3.48), (3.51) и (3.52), окончательно получаем систему уравнений, задающих закон функционирования тетрасумматора $(j = 2i,\ i = \overline{0, n-1})$:

$$\begin{cases} s_j = x_j \oplus y_j \oplus c_{j-2}; \\ c_j = x_j y_j \vee x_j c_{j-2} \vee y_j c_{j-2}; \\ s_{j+1} = x_{j+1} \oplus y_{j+1} \oplus c_{j-1}; \\ c_{j+1} = x_{j+1} y_{j+1} \vee x_{j+1} c_{j-1} \vee y_{j+1} c_{j-1}. \end{cases} \quad (3.53)$$

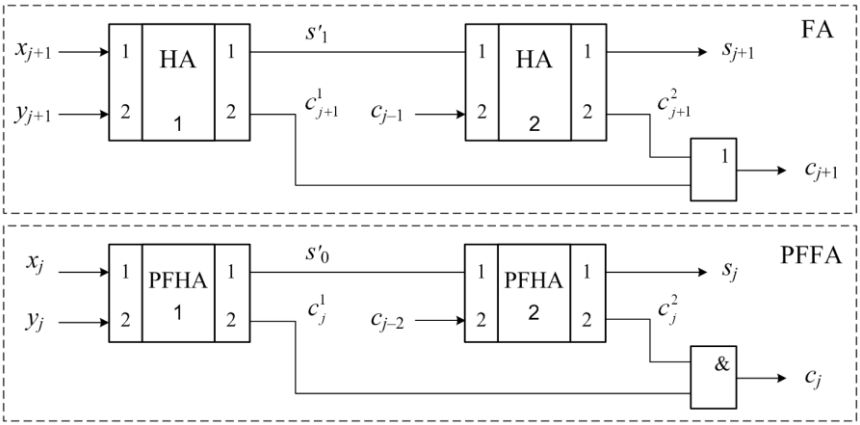


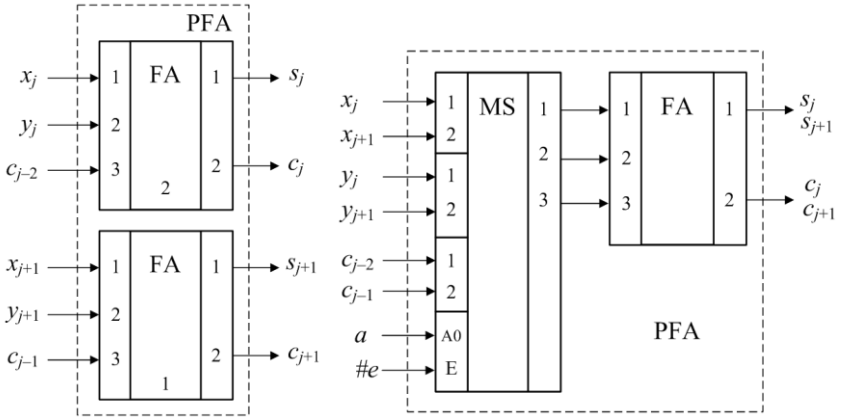
Рисунок 3.15 — Разбиение схемы реализации тетрасумматора с выделением блоков FA (Full Adder) и PFFA (Postbinary Fraction Full Adder)

На основании полученных результатов можно предположить, что для постбинарного сумматора возможны две реализации:

1. PFA = 2FA — реализация тетрасумматора PFA на базе двух двоичных сумматорах FA (рис. 3.16 *а*).
2. PFA = MS + FA — реализация тетрасумматора PFA на базе одного сумматора FA с переключением входов при помощи мультиплексора MS (рис. 3.16 *б*).

В схеме PFA = MS + FA присутствуют сигналы адреса *a* и разрешения (стробирования) *#e* для работы

мультиплексора MS. Такая организация усложняет алгоритмы устройства управления при выполнении сложения тетракодов. Кроме того, с точки зрения быстродействия и уменьшения аппаратурных затрат, схема PFA = 2FA является предпочтительной.



а) *б*)

Рисунок 3.16 — Реализации тетрасумматора PFA с использованием двоичных компонентов

Таким образом, схему PFA = 2FA можно построить на двух одноразрядных двоичных сумматорах, схема которых является наиболее экономичной [8, с. 94–95]. Закон функционирования таких сумматоров задается уравнениями:

$$\begin{cases} s_j = \overline{\overline{s^0_{\text{int}} s^0_{\text{int}} c_{j-2}} \cdot \overline{c_{j-2} s^0_{\text{int}} c_{j-2}}}; \\ c_j = \overline{\overline{x_j y_j} \cdot \overline{s^0_{\text{int}} c_{j-2}}}; \\ s_{j+1} = \overline{\overline{s^1_{\text{int}} s^1_{\text{int}} c_{j-1}} \cdot \overline{c_{j-1} s^1_{\text{int}} c_{j-1}}}; \\ c_{j+1} = \overline{\overline{x_{j+1} y_{j+1}} \cdot \overline{s^1_{\text{int}} c_{j-1}}}, \end{cases} \tag{3.54}$$

где $s^0_{\text{int}} = \overline{x_j \overline{x_j y_j} \cdot y_j \overline{x_j y_j}}$, $s^1_{\text{int}} = \overline{x_{j+1} \overline{x_{j+1} y_{j+1}} \cdot y_{j+1} \overline{x_{j+1} y_{j+1}}}$, $j = 2i$ ($i = \overline{0, n-1}$).

Система уравнений (3.54) позволяет получить схему на рис. 3.17.

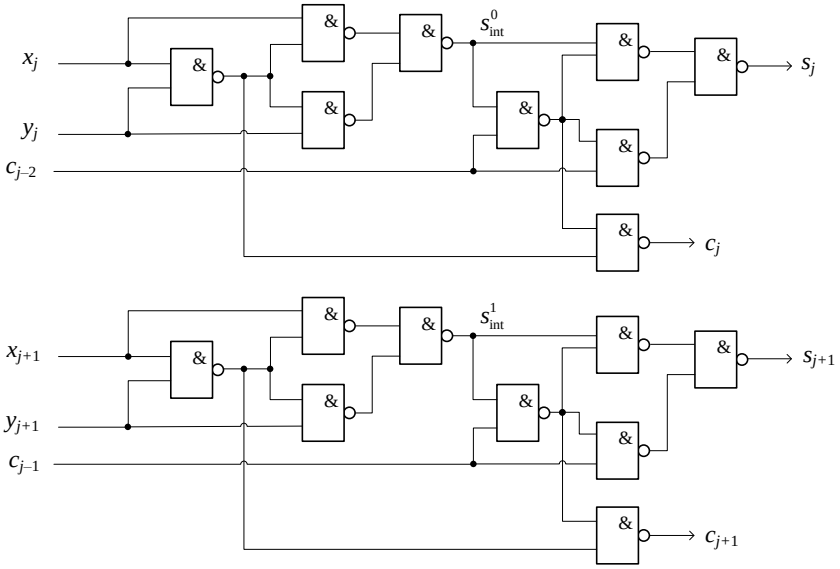
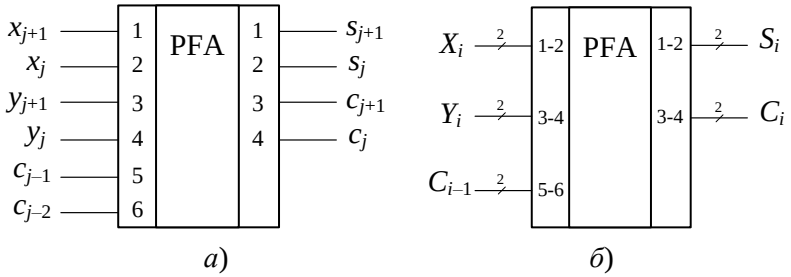


Рисунок 3.17 — Схема однотетритного постбинарного сумматора PFA

Схема из рис. 3.17 использует 18 элементов И-НЕ и имеет цену по Квайну, равную 36. При этом, поскольку двоичные сумматоры работают параллельно, временные параметры тетрасумматора равны аналогичным параметрам двоичного сумматора: $t_C = 5t_G$, $t_S = 6t_G$.

Условное обозначение однотетритного постбинарного сумматора PFA приведено на рис. 3.18.



а) *б*)

Рисунок 3.18 — Условное графическое обозначение тетрасумматора PFA: с двоичными входами/выходами (*а*) и обобщенное для тетритов (*б*)

В результате синтеза постбинарных суммирующих компонентов выяснилось, что они могут быть частично (для постбинарного полусумматора) или полностью (для тетрасумматора) замещены двоичными эквивалентами. При этом постбинарные суммирующие компоненты обладают тем же временем суммирования при неизбежном росте аппаратных затрат (табл. 3.8).

На рис. 3.19 представлена упрощенная схема каскадирования тетрасумматоров для сложения двух *n*-разрядных тетракодов *X* и *Y*.

Таблица 3.8 — Сравнительные характеристики постбинарных и двоичных суммирующих схем

Тип и обозначение суммирующего устройства		*Цена по Квайну*	*Быстродействие*	
			t_S	t_C
Полусумматоры	HA	14	$3t_G$	$2t_G$
	PHA	24		
Сумматоры	FA	18	$6t_G$	$5t_G$
	PFA	36		

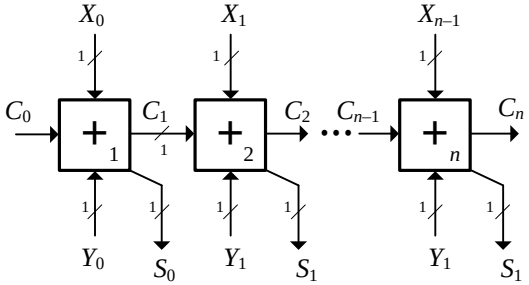
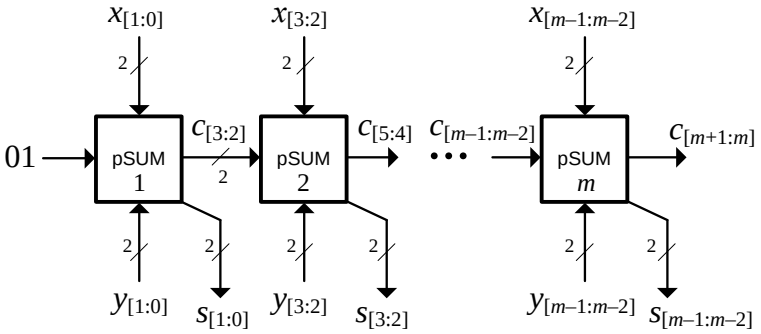


Рисунок 3.19 — Упрощенная схема каскадирования тетрасумматоров для сложения двух *n*-разрядных тетракодов *X* и *Y*

Используя работающие с двоичным кодом компоненты pSUM, можно получить m-разрядный сумматор, выполняющий функцию сложения и вычитания тетракодов в двоичной записи *x* и *y*. В качестве входных переносов в случае сложения подается пара бит «01» (эквивалентно тетриту «0»), а в случае вычитания — «10» (эквивалентно тетриту «1»). Схемы сложения и вычитания тетракодов в *m*-разрядных двоичных кодах представлены на рис. 3.20.

а) $s = x + y$



б) $s = x - y$

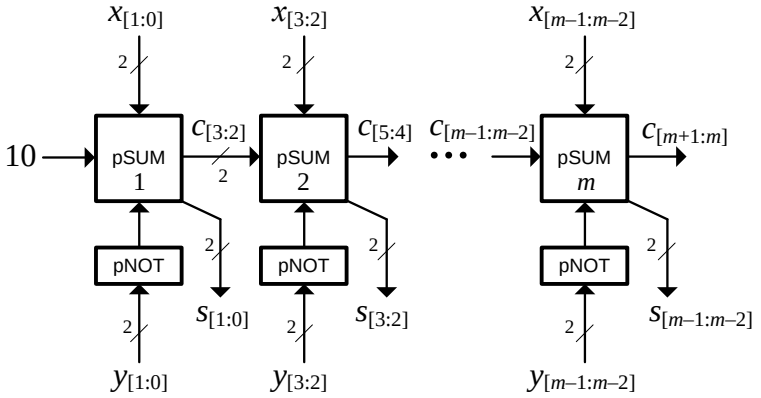
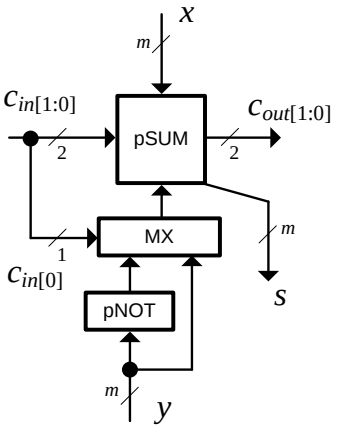


Рисунок 3.20 — Схемы сложения (а) и вычитания (б) тетракодов в m-разрядной двоичной записи

На рис. 3.21 представлена схема, обобщающая представленные на рис. 3.20 схемы сложения и вычитания тетракодов в m-разрядной двоичной записи (для n-разрядных тетракодов при $n = m/2$).



$$c_{in} = \begin{cases} 01: \ s = x + y; \\ 10: \ s = x - y; \\ 00, 11: \ error. \end{cases}$$

Рисунок 3.21 — Обобщенная схема сложения
и вычитания тетракодов в *m*-разрядной двоичной записи

3.3. Аппаратная реализация умножения тетракодов

Алгебраическое умножение тетритов подобно логическому умножению состояний тетралогики согласно табл. 2.9. Поэтому целесообразно использовать логический элемент pOR для организации поразрядного умножения тетракодов.

На рис. 3.22 представлена функциональная схема умножения *n*-разрядных тетракодов *X* и *Y*, представленных *m*-разрядными ($m = 2n$) двоичными записями x и y с получением $2m$-разрядной двоичной суммы s.

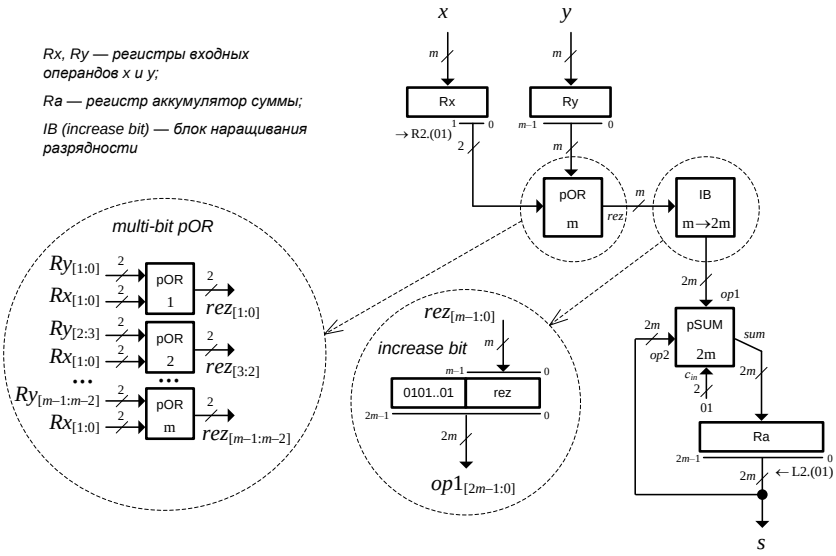


Рисунок 3.22 — Функциональная схема умножения тетракодов в *m*-разрядной двоичной записи

В функциональной схеме умножения тетракодов регистры Rx и Ra способны выполнять логические сдвиги их содержимого на 2 разряда вправо (R2.01) и влево (L2.01) соответственно с занесением в освободившиеся разряды двоичных значений «01» (значение тетрануля). Связка 2m-разрядных блоков pSUM и Ra выполняют функцию аккумулирования: происходит накапливание суммы *m* слагаемых, подаваемых последовательно на вход *op*1.

На приведенном рис. 3.23 изображена блок-схема выполнения операции умножения тетракодов, представленных *m*-разрядными двоичными кодами. Реализация алгоритма умножения невозможна без блока управления, который выполнит анализ и

последовательность необходимых шагов вычислений, организованных в операционном блоке.

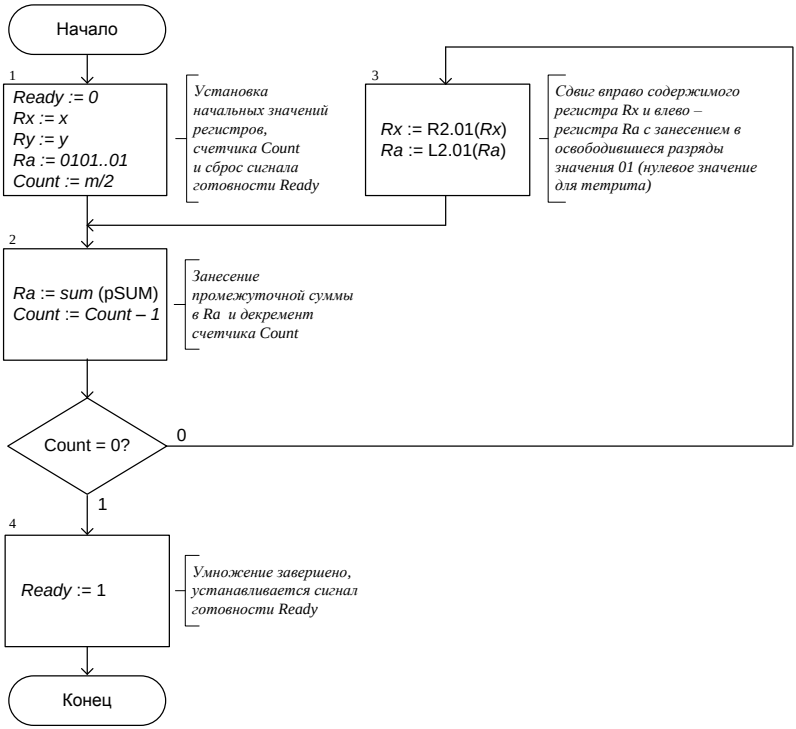


Рисунок 3.23 — Алгоритм умножения тетракодов, представленных *m*-разрядными двоичными кодами

На рис. 3.24 приведена функциональная схема умножения тетракодов, состоящая из блока управления U1 (MPU_for_MUL) и операционного блока U2 (pMUL) со следующим обозначением сигналов:

1) на входе:

- A, B — шины входных данных разрядностью bus_with + 1;
- Reset — сигнал сброса: для MPU_for_MUL — возврат к исходному состоянию; для pMUL — обнуление содержимого всех регистров;
- Start — сигнал начала работы блока (начало умножения);
- clk — синхросигнал (для моделирования использовались сигналы в диапазоне частот 1–10 ГГц).

2) на выходе:

- S — шина выходных данных разрядностью 2·bus_with + 1;
- tDone — сигнал завершения итерации на блоке pMUL (завершение получения промежуточной суммы на очередном шаге);
- RDY — сигнал готовности схемы: RDY = 1 свидетельствует о завершении операции умножения и готовности принять следующую пару значений.

Блок MPU_for_MUL управляет блоком pMUL через трехбитную шину Command [2:0], значения которой сопоставляются управляющим сигналам, значение и описание которых приведены в табл. 3.9, а граф состояний управляющего блока на рис. 3.25.

На рис. 3.26 приведена диаграмма состояний при выполнении умножения 4-разрядных тетракодов, записанных 8-разрядными двоичными кодами.

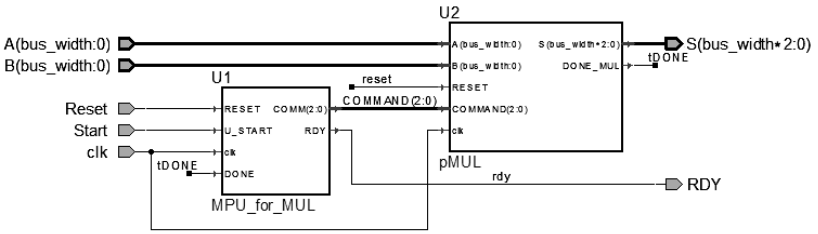


Рисунок 3.24 — Функциональная схема блока умножения тетракодов

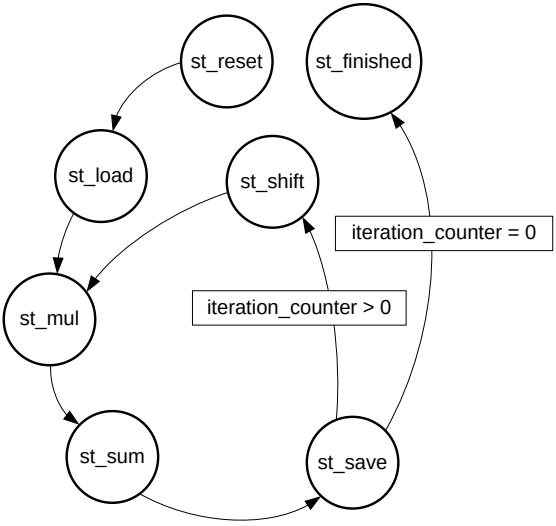


Рисунок 3.25 — Граф состояний управляющего блока MPU_for_MUL схемы умножения тетракодов

Из рис. 3.25 следует, что если A = 10101010b = 1111t и B = 01011100b = MAt, то $S = A \cdot B$ = 0101011110101000b = M111At. При декодировании получаем

$A \rightarrow$ [1111, 1111] = [15, 15],

$B \rightarrow [01, 10] = [1, 2]$ и $S \rightarrow [01111, 11110] = [15, 30]$,

что соответствует умножению интервалов [15, 15] и [1, 2] по формуле

$$[x_1, x_2] \times [x_3, x_4] =$$

$$= [\min(x_1x_3, x_1x_4, x_2x_3, x_2x_4), \max(x_1x_3, x_1x_4, x_2x_3, x_2x_4)].$$

с получением результирующего интервала [15, 30].

Таблица 3.9 — Управляющие сигналы для блока pMUL

Значение Command	*Название состояний*	*Описание*
000	st_reset	Обнуление регистров и счетчика
001	st_load	Загрузка входных значений, установка счетчика
010	st_mul	Ожидание умножения на текущем шаге цикла
011	st_sum	Ожидание сложения на текущем шаге цикла
100	st_save	Сохранение промежуточных данных в аккумулятор Ra
101	st_shift	Выполнение сдвиговых операций регистров Rx и Ra
110	st_finished	Окончание цикла умножения, установка RDY = 1
111	–	Резерв

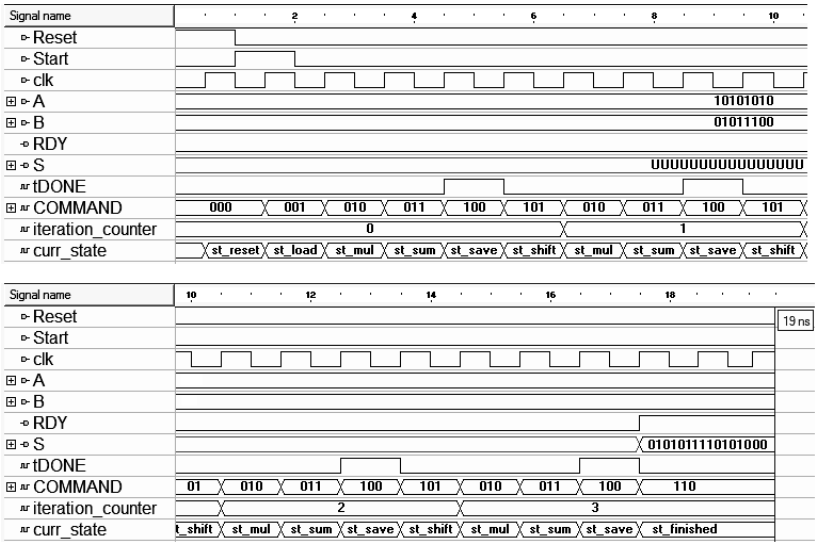


Рисунок 3.26 — Диаграмма состояний блока умножения с демонстрацией получения результата умножения 4-разрядных тетракодов, записанных в 8-разрядных двоичных кодах

3.4. Программная реализация постбинарного кодирования интервалов

На рис. 3.27, 3.28 представлены два варианта реализации программы, которая позволяет декодировать 32-разрядный тетракод с учетом описанных во второй главе (п. 2.7) преобразований.

Программа позволяет выполнить преобразование тетракодов в двоичные и десятичные наборы чисел. Исходный тетракод может быть «целочисленным» (сводиться к двоичным 32-разрядным целым беззнаковым

числам) либо «вещественным» (сводиться к двоичным 32-разрядным числам с плавающей запятой) [13, 14].

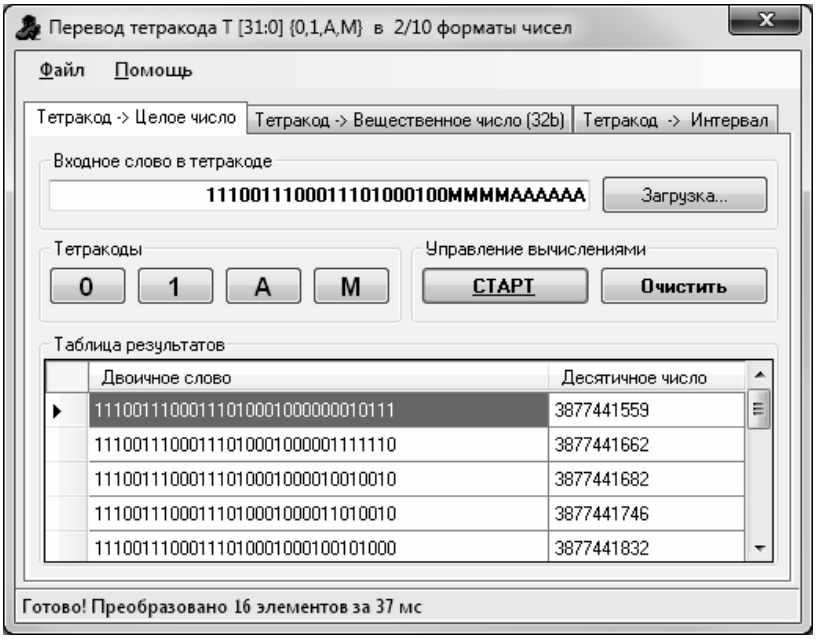


Рисунок 3.27 — Вариант программы, реализованной на языке C# (результат работы режима «Тетракод → Целое число»)

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

1) *Тетракод → Целое число:* преобразование «целочисленного» тетракода в набор целых 32-разрядных чисел (рис. 3.27).
2) *Тетракод → Целочисленный интервал:* сведение 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:* сведение тетракода к паре 32-разрядных вещественных чисел, являющихся границами интервала (рис. 3.28).

Рисунок 3.28 — Вариант программы, реализованной на языке ActionScript (результат работы режима «Тетракод → Вещественный интервал»)

Программа имеет «интеллектуальный» механизм ввода значений, обеспечивающий «комфортный» ввод данных (ввод данных с клавиатуры и с экрана, возврат на предыдущий шаг ввода, очистка поля и др.) и не позволяющий работать с некорректно введенной информацией (блокировка работы при событиях «введен не тетракод» для всех режимов и «некорректное

размещение тетритов A и M в тетракоде» для режимов 2 и 4). 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.

В программе предусмотрен ввод предустановленных значений тетракода (рис. 3.28, поле «Выбор примеров»), а также загрузка из файла заранее подготовленных значений (рис. 3.27, кнопка «Загрузка…»). Имеется мультиязычная поддержка (русский, украинский и английский языки) интерфейса и справочных материалов.

Программа занимает около 250 КБ (flash-приложение) и 60 КБ (C#-приложение) дискового пространства. Вариант программы на flash отличается ярким динамическим интерфейсом, и изначально планировался для использования в виде веб-приложения.

Рассмотрим декодирование тетракода на примере работы программы в 3-м и 4-м режимах. Значения тетритов исходного тетракода *T* представлены на рис. 3.29.

Рисунок 3.29 — Поле ввода вещественного тетракода flash-приложения (элементы checkbox определяют доступность полей к вводу значений)

Результаты работы программы в режиме 3 (сведение тетракода к набору вещественных чисел) представлены на рис. 3.30.

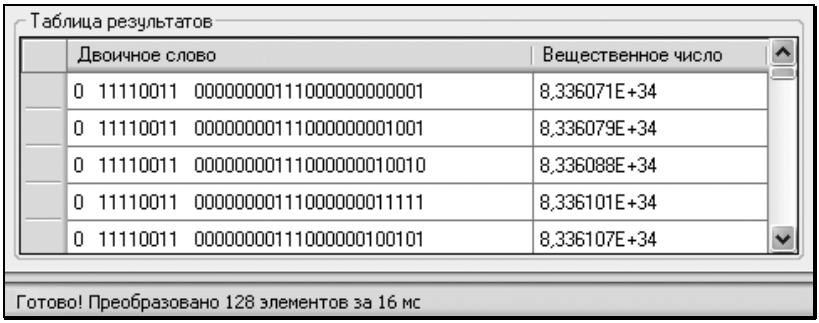


Рисунок 3.30 — Результат сведения тетракода к набору вещественных чисел

За 16 мс программа выполнила сведение тетракода T к набору из $2^8 = 128$ вещественных чисел (в тетракоде 8 тетритов М). При этом тетриты М в своей совокупности обеспечили полный перебор двоичных значений, а тетриты А 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 А каждый раз будут заполняться случайными двоичными значениями.

В 4-м режиме работы программы (сведение тетракода к вещественному интервалу) возможны модификации получения результата (см. рис. 3.28): получение диапазонов колебаний границ («минимальная ширина интервала», «максимальная ширина интервала») и получение интервала с учетом случайного замещения двоичными значениями тетритов А. При отсутствии тетритов А в тетракоде, программа будет возвращать

одинаковые интервалы не зависимо от выбранной модификации режима.

На рис. 3.31 представлены результаты работы программы при сведении тетракода T к интервальным границам в каждой модификации.

а)

Настройка вывода

○ Макс. ширина (wid) ○ Мин. ширина (wid) ○ Случайные границы

Таблица результатов

№	Двоичное слово	Вещественное число
1	0 11110011 00000000111000000000000	8,336070E+034
2	0 11110011 00000000111001111111111	8,337083E+034

[8,33607E+34; 8,337083E+34] wid = 1,01313E+31; mid = 8,336577E+34

б)

Настройка вывода

○ Макс. ширина (wid) ◉ Мин. ширина (wid) ○ Случайные границы

Таблица результатов

№	Двоичное слово	Вещественное число
1	0 11110011 00000000111000000000111	8,336077E+034
2	0 11110011 00000000111001111111000	8,337077E+034

[8,336077E+34; 8,337077E+34] wid = 9,992652E+30; mid = 8,336577E+34

в)

Настройка вывода

○ Макс. ширина (wid) ○ Мин. ширина (wid) ◉ Случайные границы

Таблица результатов

№	Двоичное слово	Вещественное число
1	0 11110011 00000000111000000000101	8,336075E+034
2	0 11110011 00000000111001111111110	8,337082E+034

[8,336075E+34; 8,337082E+34] wid = 1,007188E+31; mid = 8,336579E+34

Рисунок 3.31 — Результат сведения тетракода к вещественному интервалу (*а* — получение «крайне широких» границ; *б* — получение «крайне узких» границ, *в* — получение «случайных» границ)

Модификацию «максимальная ширина» (рис. 3.31 *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 (табл. 3.10) для левой и правой границы соответственно, т. е. [MIN_M(MIN_A(*T*)); MAX_M(MAX_A(*T*))] — искомый интервал.

Таблица 3.10 — 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

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

Модификация «минимальная ширина» (рис. 3.31 *б*) отображает «худшие» случаи относительно тетритов A при определении границ интервала. Границы интервала

«минимальной ширины» из тетракода T можно получить, используя соотношения (2) и (3) для левой и правой границы соответственно, т. е.

[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 сведенным из тетракода интервалом.

Действительно, границы интервала (с учетом разрядов A), полученные при модификации «случайные границы» (рис. 3.31 *в*), никогда не будут меньше по модулю соответствующих границ интервала «минимальной ширины». Таким образом, границы сведенного из тетракода T интервала при любом случайном наборе бит для разрядов A будут принадлежать следующим диапазонам:

[MIN_M(MIN_A(T)); MIN_M(MAX_A(T))] — для левой границы;

[MAX_M(MIN_A(T)); MAX_M(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

$$\left[\begin{matrix}\left[\text{MIN\_M}(\text{MIN\_A}(T));\ \text{MIN\_M}(\text{MAX\_A}(T))\right];\\ \left[\text{MAX\_M}(\text{MIN\_A}(T));\ \text{MAX\_M}(\text{MAX\_A}(T))\right]\end{matrix}\right] \Rightarrow$$

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

где Rnd_A — функция, сводящая тетрит A к двоичным 0 или 1, которые выбираются случайным образом.

3.5. Оценка погрешности представления вещественных чисел в постбинарных форматах с плавающей запятой

Постбинарные форматы чисел с плавающей запятой (pbinary, или pb) впервые были описаны в [23] как модификация существующих форматов стандарта IEEE 754–2008. Фактически, **постбинарный формат** — это формат с плавающей запятой, поддерживающий постбинарное кодирование и содержащий на месте младших разрядов мантиссы служебную информацию, определяющую точность и тип числа, представленного в данном формате. Разработка постбинарных форматов основывается на преодолении недостатков существующих стандартных форматов.

Актуальность перехода от бинарных форматов к постбинарным обусловлена, прежде всего, существующими и неустранимыми недостатками современного представления вещественных чисел в форматах с плавающей запятой и выполнением арифметических операций над ними. Ключевыми недостатками бинарных форматов являются как неизбежные округления чисел (при кодировании и во время вычислений), так и отсутствие информации о точности величины, представленной битовым полем формата. Пример того, как эти недочеты приводят к получению неверных результатов, был продемонстрирован З. Румпом, представившим полином (названный по фамилии автора — полином Румпа), который при определенном сочетании значений переменных дает заведомо неправильный результат [7, с. 173]. Постбинарные форматы лишены подобных недостатков, поскольку, во-первых, ориентированы на динамическое

наращивание разрядности, во-вторых, способны работать с интервальными и дробными значениями, и, в-третьих, использование постбинарного кодирования дает возможность избежать грубых округлений и потерь значимых разрядов мантиссы.

В работах [2, 7, 15, 23] авторами подробно рассмотрены **постбинарные форматы чисел с плавающей запятой** (рис. 3.32). Разработка данных форматов основывается на недостатках существующих. Поскольку произвольное вещественное число представляется бесконечной систематической (например, десятичной или двоичной) дробью, то на практике в научных и инженерных вычислениях вещественные числа приходится представлять в компьютере конечными дробями, чаще всего числами с плавающей запятой (числами, представленными в формате IEEE 754). Следовательно, числа с плавающей запятой представляют конечное множество, на которое отображается бесконечное множество вещественных чисел. Поэтому исходное число может быть представлено в формате IEEE 754 не точно. Это один из ряда недостатков чисел формата IEEE 754, которые подробно рассмотрены в [2, 16, 17].

Однако следует сделать акцент на обозначении форматов (см. Приложение): постбинарный формат определен словом «pbinary» или сочетанием «pb», после которого стоит цифра, определяющая длину поля (в битах), занимаемую форматом. Следующая цифра (отделенная от первой слешем «/») определяет разрядность тетракода или составного слова (дроби или интервала), «упакованного» в основном формате. Появление второй цифры требует уточнения, за которое отвечает

своеобразный «суффикс» формата: p — постбинарное число (тетракод); f — дробь; i — интервал.

Например:

1) формат pbinary128 представляет собой число счетверенной точности (формат представлен 128 битами);
2) формат pb256/64fp — постбинарный формат, в 256-ти двоичных разрядах которого хранятся два 64-разрядных числа — числитель и знаменатель дроби (на что указывает «f»), при этом каждое 64-разрядное число занимает 128 двоичных разрядов, поскольку является постбинарным (на что указывает «p») каждый разряд которого (тетрит) кодируется парой битов. Следовательно, 2 поля по 64 разряда тетракода занимают $2 \times (2 \times 64) = 256$ бит.

Особенности реализации постбинарных форматов, способы представления чисел этими форматами, оценки погрешностей и полное описание служебных и информационных полей подробно рассмотрены в [7, с. 196–217].

С целью увеличения точности преставления чисел, а также повышения надежности вычислений, для чисел с плавающей точкой были предложены постбинарные форматы чисел от одинарной до квадроточности [2, с. 202] (см. Приложение).

Используемый в постбинарных форматах способ кодирования данных основан на принципах кодологического базиса [7, 18, 19]: в качестве кодовой системы выступает тетракод *T*, а в качестве единицы хранения одного разряда — тетрит *t* (п. 2.2), кодирующий одно из четырех значений: двоичный ноль (0), двоичную единицу (1), значение неопределенности (A), значение

множественности (M). При кодировании числовых значений, тетрит *t* = A может принимать любое (случайное) значение 0 или 1, а тетрит *t* = M — и 0 и 1 одновременно (т. е. представлять два числовых набора).

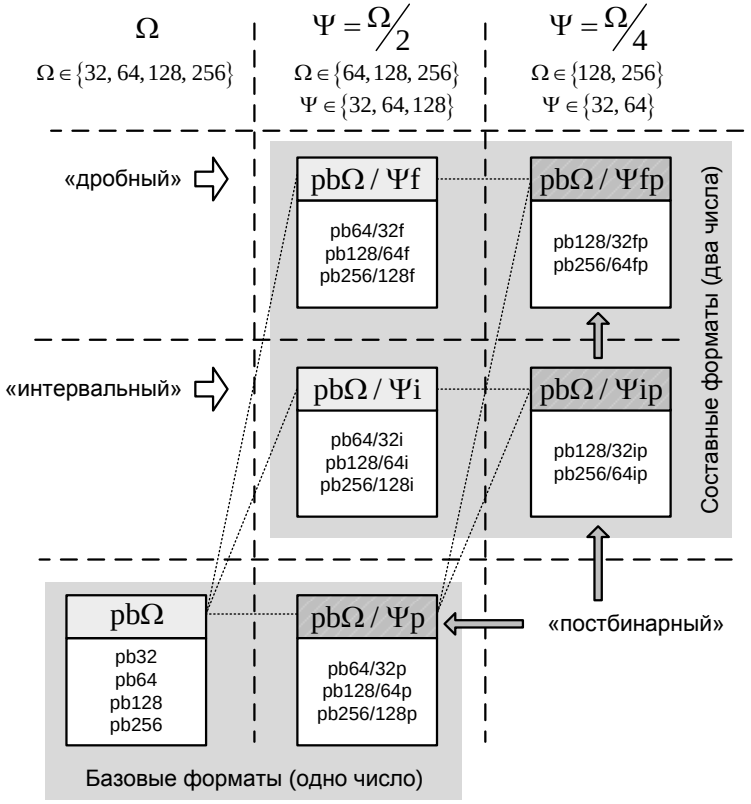


Рисунок 3.32 — Соотношение модифицированных форматов с указанием способов кодирования и назначения (pb – сокращенное обозначение постбинарного формата «pbinary»)

Такое «гибкое» кодирование количественных значений позволяет с высокой степенью точности представлять числа в форматах 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

$$P_\Omega = 2^\Omega = 2^{s+l+m}, \qquad (3.55)$$

где P_Ω — количество чисел в формате с плавающей запятой, представленное Ω-разрядным двоичным значением; s, l, m — разрядности знака, порядка и мантиссы соответственно.

Имея в арсенале конечное количество кодируемых в формате с плавающей запятой значений (точек разрядной сетки или **базовых точек**) вещественной оси, вся дальнейшая процедура представления любого вещественного числа сводится к отображению его одной (чаще всего ближайшей) базовой точкой. 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 $z \in \boldsymbol{Z}$ («кратность отрицательной степени двойки»).

Остальные числа представлены в формате с плавающей запятой приближенно, т. е. с некоторой ошибкой точности.

На рис. 3.33 приведен график ошибки точности (погрешности) представления числа в формате IEEE 754–2008 при увеличении порядка на 1 (т. е. от k до k+1). При этом базовые точки n_i — фактически разрядная сетка поля мантиссы формата с плавающей запятой (на рисунке — светлые точки).

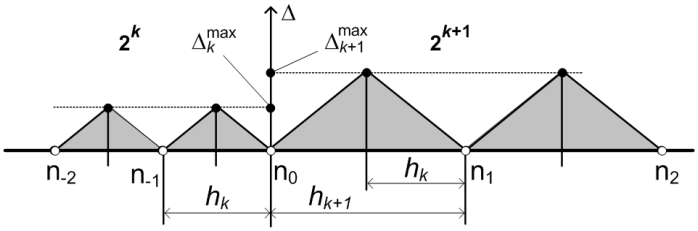


Рисунок 3.33 — График ошибки точности представления чисел в формате IEEE 754–2008

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_k между соседними базовыми точками, т. е. позициями чисел представленных полями форматов с единым десятичным значением порядка k ($k \neq 0$) и с различающимися в один бит мантиссами можно определить по формуле:

$$h_k = 2^{E_k - off - m}, \tag{3.56}$$

где *off* — десятичное смещение порядка *E*; *m* — число разрядов поля мантиссы; E_k — десятичное число, полученное из поля порядка двоичного числа с плавающей запятой (значение смещенного порядка).

Расстояние h_k — это фактически шаг чисел одного порядка, который удваивается с увеличением порядка числа с плавающей запятой на единицу:

$$h_{k+1} = 2 \cdot h_k. \tag{3.57}$$

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

Следует отметить, что формулы (3.56) и (3.57) справедливы для расчета шага нормализованных чисел [20]. Для денормализованных чисел (при $E_k = 0$ и ненулевой мантиссе) примем значение $k = 0$, показывая что такие числа близки к нулю. Шаг h_0 для множества денормализованных чисел можно определить следующим образом:

$$h_0 = 2^{1 - off - m}. \tag{3.58}$$

Очевидно, что h_0 — минимальный шаг, равный по своему значению минимальному денормализованному числу.

Абсолютная максимальная ошибка для нормализованных ($\Delta_k^{\max}$) и денормализованных ($\Delta_0^{\max}$) чисел в формате IEEE 754–2008 равна в пределе половине шага:

$$\Delta_k^{\max} = \frac{1}{2} h_k = \frac{1}{2}\left(2^{E_k - off - m}\right) = 2^{E_k - off - m - 1}; \tag{3.59}$$

$$\Delta_0^{\max} = \frac{1}{2} h_0 = \frac{1}{2}\left(2^{1 - off - m}\right) = 2^{-(off + m)}. \tag{3.60}$$

Учитывая (3.57) и (3.59), для нормализованных чисел можно получить зависимость

$$\Delta_{k+1}^{\max} = 2 \cdot \Delta_k^{\max}, \tag{3.61}$$

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

Относительная максимальная ошибка нормализованных ($\delta_k^{\max}$) и денормализованных ($\delta_0^{\max}$) чисел в формате IEEE 754–2008 равна

$$\delta_k^{\max} = \frac{\Delta_k^{\max}}{|F_k|} \cdot 100\%; \tag{3.62}$$

$$\delta_0^{\max} = \frac{\Delta_0^{\max}}{|F_0|} \cdot 100\%, \tag{3.63}$$

где F_k и F_0 — десятичные числа, полученные из форматов представления нормализованных и денормализованных чисел с плавающей запятой соответственно:

$$|F_k| = 2^{E_k - off} \cdot \left(1 + \frac{M_n}{2^m}\right); \tag{3.64}$$

$$|F_0| = 2^{1-off} \cdot \left(\frac{M_n}{2^m}\right), \tag{3.65}$$

где M_n — десятичное число, полученное из m-разрядного поля мантиссы двоичного числа с плавающей запятой ($0 < n < 2^m$).

Окончательно получаем

$$\delta_k^{\max} = \frac{1}{2^{m+1} + 2 \cdot M_n} \cdot 100\%; \tag{3.66}$$

$$\delta_0^{\max} = \frac{1}{2 \cdot M_n} \cdot 100\%, \qquad (3.67)$$

Для наглядности, формулы нахождения значений максимальной абсолютной и относительной погрешностей представления нормализованных и денормализованных чисел для бинарных и постбинарных форматов сведены в табл. 3.11. Отличия в полученных числовых значениях погрешностей для Ω-разрядных бинарных (binaryΩ) и аналогичных им постбинарных форматов (pbinaryΩ), связаны, прежде всего, с организацией полей постбинарного формата, разрядность мантиссы *m* которого меньше на число разрядов *id* поля идентификатора формата (см. табл. 3.11 и [7, с. 203 – рис. 4.16]).

Таким образом, величины относительной ошибки δ_k числа, представленного в формате binaryΩ и относительной ошибки $\delta_k^{/}$ числа, представленного в аналогичном ему формате pbinaryΩ (т. е. форматы с равными порядками E_k $(k \geq 0)$ и разрядностью Ω) связаны следующим соотношением:

$$\delta_k^{/} = 2^{id} \cdot \delta_k, \qquad (3.68)$$

где *id* — разрядность идентификатора формата pbinaryΩ, являющаяся фактически разницей между разрядностями мантисс форматов binaryΩ и pbinaryΩ.

Из формулы (3.68) следует, что за счет «укороченной» мантиссы формат pbinaryΩ имеет абсолютную ошибку представления числа, большую в 2^{id} раза, чем у формата binaryΩ.

Таблица 3.11 — Формулы максимальной абсолютной и относительной ошибок представления нормализованных и денормализованных чисел для бинарных (binary) и постбинарных (pbinary) форматов]

Формат	*off*	*m*	*id*	$\Delta_0^{\max}$	$\Delta_k^{\max}$	$\delta_k^{\max}$, %
binary16*	15	10	–	2^{-25}	2^{E_k-26}	$(2^{11}+2M_n)^{-1}\cdot 100\%$
pbinary16	15	9	1	2^{-24}	2^{E_k-25}	$(2^{10}+2M_n)^{-1}\cdot 100\%$
binary32	127	23	–	2^{-150}	2^{E_k-151}	$(2^{24}+2M_n)^{-1}\cdot 100\%$
pbinary32	127	21	2	2^{-148}	2^{E_k-149}	$(2^{22}+2M_n)^{-1}\cdot 100\%$
binary64	1023	52	–	2^{-1075}	2^{E_k-1076}	$(2^{53}+2M_n)^{-1}\cdot 100\%$
pbinary64	1023	48	4	2^{-1071}	2^{E_k-1072}	$(2^{49}+2M_n)^{-1}\cdot 100\%$
binary128	16383	112	–	2^{-16495}	$2^{E_k-16496}$	$(2^{113}+2M_n)^{-1}\cdot 100\%$
pbinary128	16383	104	8	2^{-16487}	$2^{E_k-16488}$	$(2^{105}+2M_n)^{-1}\cdot 100\%$
pbinary256	524287	219	16	$2^{-524506}$	$2^{E_k-524507}$	$(2^{220}+2M_n)^{-1}\cdot 100\%$

* — не входит в стандарт IEEE 754–2008, однако определен как формат чисел половинной точности.

Этот недостаток постбинарного формата с плавающей запятой компенсируется, во-первых, поддержкой динамической разрядности, при которой число в процессе вычисления может выражаться в форматах с возрастающей точностью, что приведет к уменьшению ошибки точности, и, во-вторых, за счет введения тетракода как системы кодирования в модифицированных «постбинарных» форматах pbinaryΩ/Ψp, pbinaryΩ/Ψfp и pbinaryΩ/Ψip (см. рис. 3.32) рассматривается наряду со стандартными способами и новый, модифицированный способ округления — **постбинарное округление.**

Стандарт IEEE 754–2008 предусматривает четыре способа округления чисел:

1. *К нулю* (рис. 3.34 *а*).

   При округлении к нулю нужно просто отбросить незначащие разряды числа, поэтому этот способ самый легкий в аппаратной реализации.

2. *К ближайшему числу* (к ближайшей базовой точке) (рис. 3.34 *б*).

   При округлении к ближайшему числу нужно **к мантиссе прибавить значение старшего незначащего разряда числа**. Данный способ округления в математическом сопроцессоре используется по умолчанию.

3. *К положительной бесконечности* (+∞) (рис. 3.34 *в*).

   Округление к +∞ применяется при кодировании интервальных чисел [21]. При округлении к +∞ нужно к полю мантиссы прибавить инверсное значение поля знака #*S*, поскольку:

- если число положительное (S = 0) — нужно к мантиссе **всегда прибавлять 1**;
- если число отрицательное (S = 1) — нужно просто **отбросить незначащие разряды**.

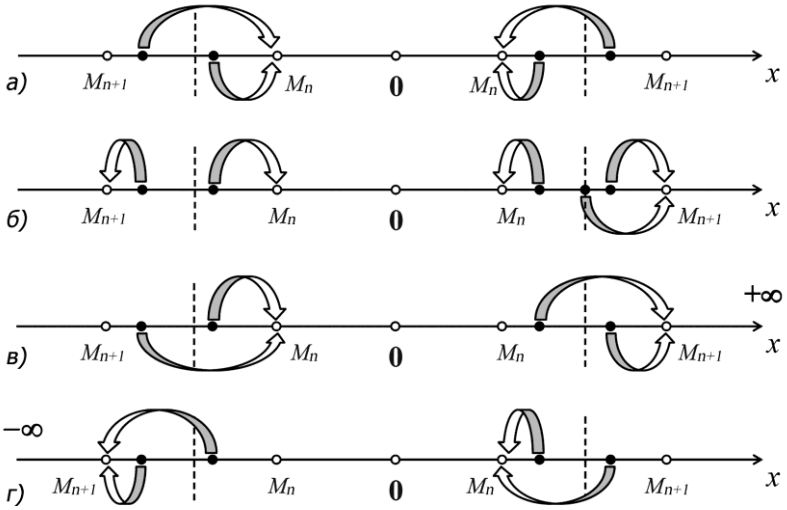


Рисунок 3.34 — Принцип округления чисел стандарта IEEE 754 (*а* – к нулю, *б* – к ближайшему числу, *в* – к положительной бесконечности, *г* – к отрицательной бесконечности)

4. *К отрицательной бесконечности* (–∞) (рис. 3.34 *г*).

   Округление к –∞ применяется при кодировании интервальных чисел. При округлении к –∞ нужно к полю мантиссы прибавить значение поля знака S, поскольку:

   - если число положительное (S = 0) — нужно просто **отбросить незначащие разряды**;

- если число отрицательное ($S = 1$) — нужно к мантиссе **всегда прибавлять 1**.

На рис. 3.34 светлые точки являются базовыми, а темные — позиции исходных чисел. Пунктирными линиями обозначена середина между соседними базовыми точками.

Таким образом, для реализации стандартных видов округления достаточно проанализировать **старший незначащий разряд двоичной дроби исходного числа**, т. е. первый разряд двоичного числа, не попавший в поле мантиссы, а также учитывать знак числа. Очевидно, что рассмотренные формулы абсолютной и относительной максимальной ошибки (3.59), (3.60) и (3.66), (3.67) справедливы только при округлении к ближайшему числу.

На рис. 3.35 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 постбинарных форматов, позволяет рассмотреть наряду со стандартными способами и новый способ округления. На рис. 3.36 приведен способ **постбинарного округления.** Причем числа, чьи позиции находятся на светлой части вещественной оси (между пунктирными линиями в участках II и III), представляются в постбинарных форматах в виде **интервального числа** с границами соседних базовых точек $[M_n, M_{n+1}]$. Такая возможность достигается появлением разрядов множественности (М) и неопределенности (А) в младших разрядах мантиссы [7, с. 155–173].

Таким образом, **постбинарное округление** — округление, оперирующее значением **¼ шага между базовыми точками**.

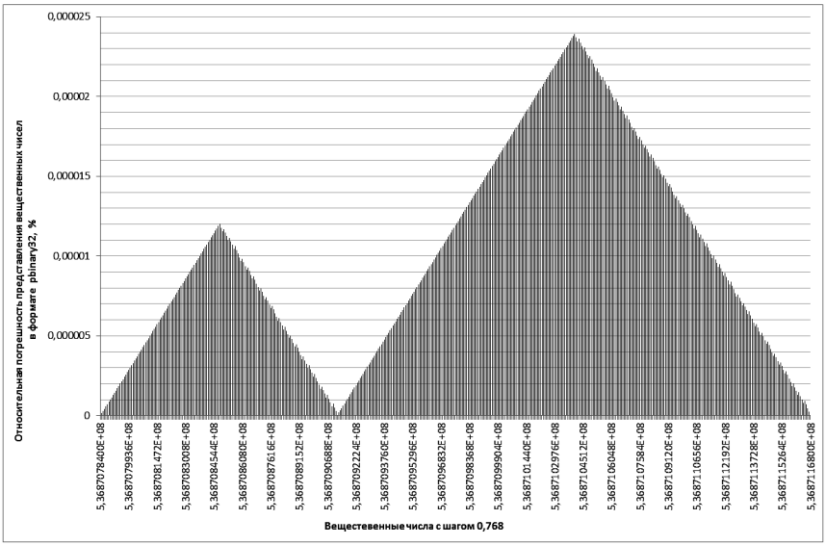


Рисунок 3.35 — Относительная погрешность вещественных чисел в формате pbinary32 при округлении к ближайшему числу

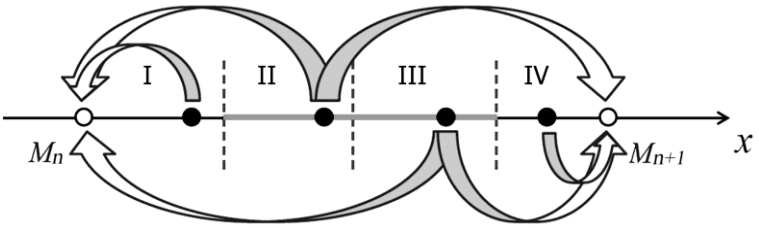


Рисунок 3.36 — Принцип постбинарного округления чисел

При постбинарном округлении **рассматриваются два старших незначащих разряда** (рис. 3.37):

- 00 — число находится в участке I вещественной оси — округление к **меньшей базовой точке** (т. е. к ближайшему числу): отбрасывание незначащих разрядов;
- 01 или 10 — число находится в участках II и III вещественной оси — формирование **нормированного тетракода** [7, с. 158–160] из поля мантиссы, т. е. фактически прибавление к мантиссе **значения множественности**;
- 11 — число находится в участке IV вещественной оси — округление к **большей базовой точке** (т. е. к ближайшему числу): прибавление 1 к полю мантиссы.

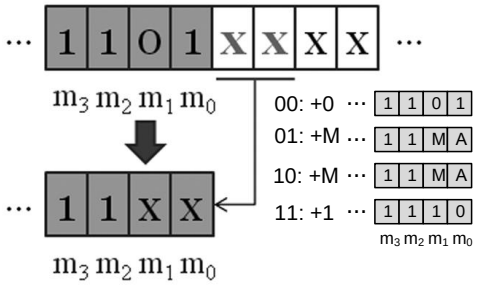


Рисунок 3.37 — Пример реализации постбинарного округления чисел для модифицированных «постбинарных» форматов с плавающей запятой

Рассмотрим пример представления числа $F = 0{,}9871625$ в формате pbinary32 с использованием

постбинарного округления. Получаем нормализованное двоичное число:

$$F_2 = 1{,}11111001011011010101\underline{\mathbf{10}}011111_2 \times 2^{-1}.$$

Здесь подчеркнуты незначащие разряды для формата pbinary32, среди которых выделены два подлежащих рассмотрению старших разряда.

Таким образом, при постбинарном округлении получаем поле формата

$$\mathrm{pbinary}(F_\mathrm{T}) = \underbrace{0}_{\text{знак}}\ \underbrace{01111110}_{\text{порядок}}\ \underbrace{11111001011011010\mathrm{MAA}}_{\text{мантисса}}\ \underbrace{00}_{\text{идентификатор}}.$$

Полученный тетракод F_T сводится к двум двоичным 32-разрядным значениям, являющимся границами интервального числа $\boldsymbol{F}$ (методы сведения тетракода к интервалу подробно рассмотрены в [7, с. 155–173]):

$$F_\mathrm{T} \to \boldsymbol{F} = [\mathrm{3F7CB6AC}, \mathrm{3F7CB6B0}]_\mathrm{h} \approx$$

$$\approx [0.98716235, 0.98716259]_{10},$$

гарантированно содержащее исходное число F.

Ширина d полученного интервала равна расстоянию между соседними базовыми точками, т. е., используя (3.56), получаем

$$d = h_k = 2^{126-127-21} = 2^{-22} =$$

$$= 2{,}384185791015625 \times 10^{-7} \approx 2{,}4 \times 10^{-7}.$$

Это же значение можно получить и как разность границ интервала:

$$d = 0{,}98716259 - 0{,}98716235 = 2{,}4 \times 10^{-7}.$$

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 ошибка $\Delta_k^{/\max}$ для этих чисел в постбинарном формате оказалась равной в пределе четверти шага чисел (рис. 3.38), что в два раза меньше аналогичной абсолютной ошибки $\Delta_k^{\max}$ для стандарта IEEE 754–2008:

$$\Delta_k^{/\max} = \frac{1}{2}\Delta_k^{\max} = \frac{1}{4}h_k = 2^{E_k - off - m - 2}. \qquad (3.69)$$

2) числа, лежащие в областях II и III вещественной оси, привести к интервалу, ширина которого равна шагу чисел постбинарного формата.

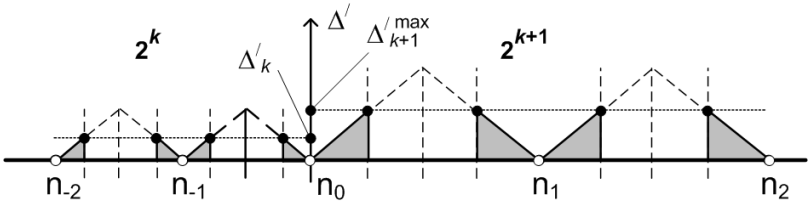


Рисунок 3.38 — График ошибки точности представления чисел в постбинарном формате

На рис. 3.39 приведена аналогичная рис. 3.35 гистограмма с определенными значениями относительной погрешности при модифицированном округлении. Действительно, по отношению к рис. 3.35, максимальные значения погрешностей уменьшились в 2 раза.

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 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 некоего 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 постбинарной арифметики, привносящей новые алгоритмы нормализации чисел, обработки исключительных ситуаций и базовых арифметических операций.

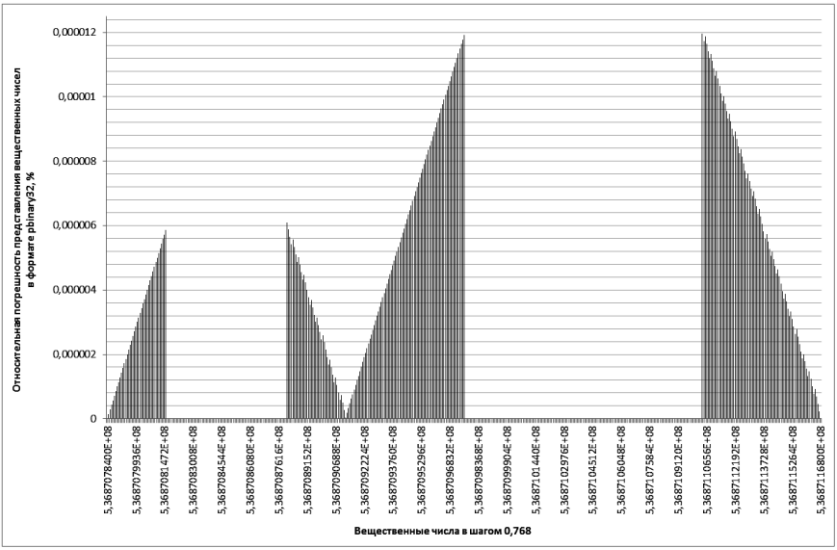


Рисунок 3.39 — Относительная погрешность вещественных чисел в формате pbinary32 при модифицированном округлении

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 [22].

3.6. Преобразователь вещественных десятичных чисел в постбинарные форматы чисел с плавающей запятой

Идея создания программного продукта, работающего с постбинарными форматами чисел с плавающей запятой, берет свое начало в целом цикле исследований, подтверждающих эффективность перехода к постбинарным форматам [7, 9, 15, 24]. 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.

Созданная **программа «Преобразователь постбинарных форматов» (Postbinary Format Converter)** 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 рис. 3.40 представлено **рабочее пространство программы Postbinary Format Converter** (далее PFC).

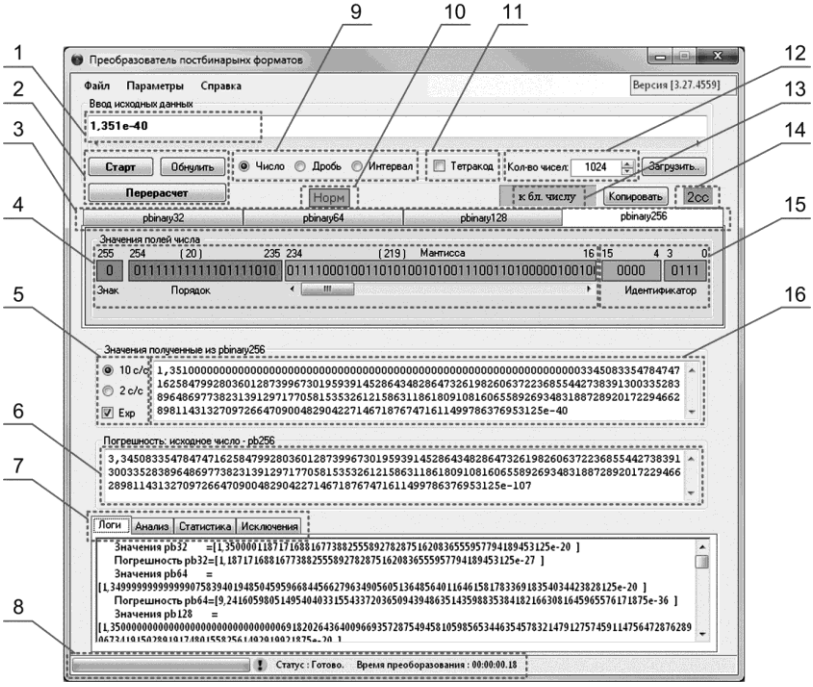


Рисунок 3.40 — Интерфейс программы PFC в режиме представления десятичного числа $1{,}351\times10^{-40}$ в 256-разрядном постбинарном формате

Основные компоненты (к.) интерфейса программы на рис. 3.40 пронумерованы следующим образом:

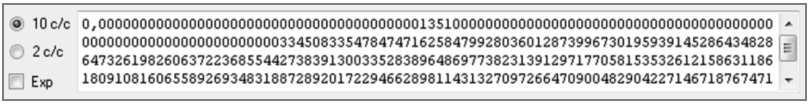
1. *Ввод исходных данных* — поле для ввода десятичного числа, подлежащего преобразованию (поддерживается ввод чисел в нормализованном виде). Возможны два варианта ввода исходных данных в текстовое поле: из клавиатуры; из файла (кнопка «Загрузить…»).

2. *Блок управления*, состоящий из трех кнопок: «Старт» — начало преобразования входного числа и извлечение его точного значения из поля формата (формирование результата в направлении *Исходное число* (к.1) → *Битовое поле формата* (к.4) → *Точное значение, полученное из поля формата* (к.16) → *Значение погрешности представления* (к.6)); «Обнулить» — обнуление всех полей, возврат в исходное состояние; «Перерасчет» — получение точного значения из отредактированного битового поля формата минуя поле исходных данных (формирование результата в направлении *Битовое поле формата* (к.4) → *Точное значение, полученное из поля формата* (к.16)).

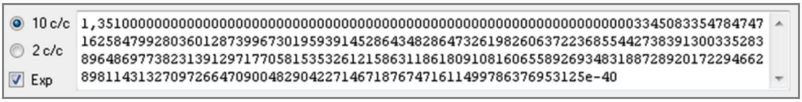
3. *Вкладки постбинарных 32-, 64-, 128- и 256-разрядных форматов*, название которых изменяется в зависимости от состояния компонентов 9 и 11 (см. табл. 3.12). Активная вкладка определяет текущий формат, в котором представлено исходное число (на рис. 3.40 это формат pbinary256).

4. *Информационное поле формата*, содержащее битовые поля знака, порядка и мантиссы с указанием номеров разрядов и количества бит в каждом поле. Информационное поле формата изменяется в зависимости от названия вкладок (к.3).

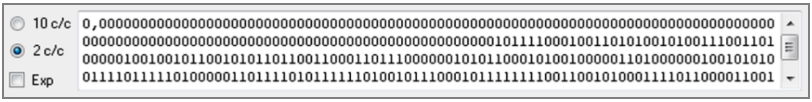
5. *Режим отображения числа* в поле компонента 16: 10с/с, 2с/с — десятичная и двоичная система счисления; Exp — нормализованный вид числа. Работа данного компонента отображена на рис. 3.41.



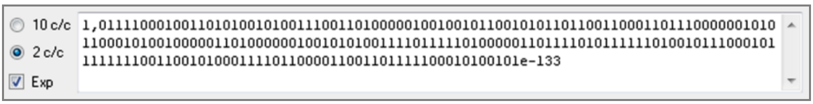
а)



б)



в)



г)

Рисунок 3.41 — Взаимодействие компонентов 5 и 16 при выводе точного значения, извлеченного из полей формата pbinary128 для входного числа 1.351e–40 (формы вывода числа: *а* – десятичное; *б* – десятичное нормализованное; *в* – двоичное; *г* – двоичное нормализованное)

6. *Значение погрешности* представления исходного числа в текущем формате (выводится всегда в нормализованном виде). Фактически это разность по модулю значений исходного числа (к.1) и точного его представления, полученного из полей текущего формата (к.16).

7. *Служебная информация* о работе программы: «Логи» — ведение истории конвертирования с возможностью ее сохранения в файле; «Статистика» — представление сравнительных характеристик (погрешности и временных затрат) для всех форматов; «Анализ» и «Исключения» — служебная информация для разработчиков (в пользовательской версии программы заменена вкладкой «Сведения о формате» — справочные данные и характеристика текущего формата).
8. *Строка состояния*, отображающая ход выполнения операции (индикатор прогресса), статуса программы, а также времени, затраченного на последнее преобразование.
9. *Переключатель формата входных данных* (табл. 3.12).
10. *Тип числа в текущем формате*: нормализованное (Норм/Norm), денормализованное (Денорм/Denorm), не число (NaN), бесконечность (Inf). Слово «Переполнение/Overflow» появляется в данном поле, если исходное число невозможно представить в текущем формате.

11. *Компонент для постбинарного кодирования*. Его включение означает, что информационные поля формата (к.4) содержат тетракод (в названии формата появляется суффикс «p») и каждая пара бит поля представляет один четверичный разряд — тетрит.

Таблица 3.12 — 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

12. *Счетчик ограничения цифр для входного числа* в поле компонента 1. Цифры входного числа, превышающие установленное количество, в дальнейших преобразованиях не участвуют и просто отбрасываются.
13. *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 специализированный способ (постбинарное округление, см. п. 3.5).

14. *Кнопка-индикатор текущей системы счисления*. Указывает и позволяет выбрать двоичную или 16-ричную систему счисления для отображения полей формата (к.4).
15. *Поле идентификатора текущего формата* — служебная часть постбинарных форматов, состоящая из полей модификатора и кода формата [7, с. 203].
16. *Поле точного значения*. Отображает результат извлечения из полей (к.4) текущего формата (к.3) без потери точности (рис. 3.41). В данном поле отображается результат преобразования двоичного числа (поля формата к.4) в десятичную дробь без ограничений вычислительных возможностей процессора и округления чисел операционной системой.

На рис. 3.42 представлена структурная модель преобразователя постбинарных форматов PFC. Ниже приведено краткое описание функциональности блоков модели.

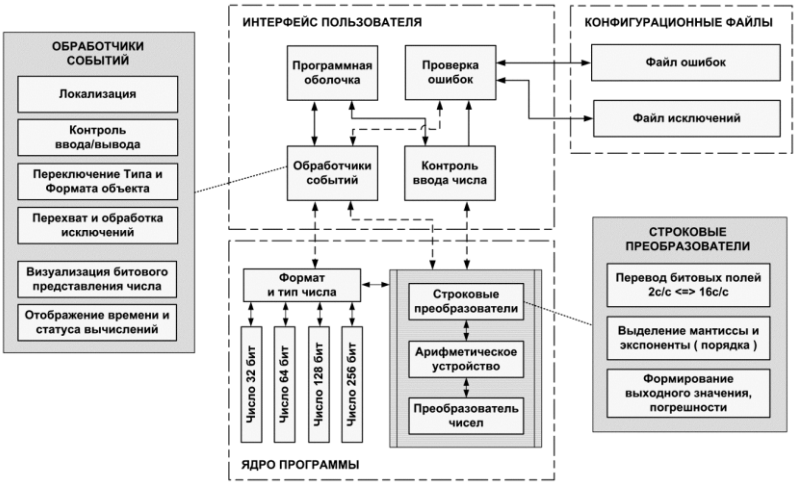


Рисунок 3.42 — Структурная модель PFC
(сплошная стрелка – передача входных и выходных параметров,
пунктирная стрелка – только выходных параметров)

Интерфейс пользователя (User Interface, UI) включает в себя набор функций, составляющих каркас программы, а также связывает все остальные блоки с основной формой программы (рис. 3.40):

- *Программная оболочка* — набор предоставляемых платформой .NET Framework классов и функций для визуализации данных в форме Windows Forms. Инкапсулирует в себе все необходимые компоненты для работы и манипуляции над данными всех типов.
- *Проверка ошибок* — блок отладки, содержащий набор функций, направленных на самопроверку и поиск ошибок в вычислениях. Запускается при изменении версии программы и непустом «Файле ошибок». Записывает возникшие в процессе работы названия исключений в «Файл исключений», а

числа в результате, которых были сгенерированы эти исключения и текущую версию в «Файл ошибок».

- *Контроль ввода числа* — набор функций, направленных на логическую обработку числа (ввод не больше одного знака «–» и «+» недопустимость ввода букв (кроме экспоненты «e» и «E») и др.) для его корректного представления.
- *Обработчики событий* — набор стандартных «слушателей событий» (event listeners), которые прикреплены к какому-то событию. Генерацию связывания объектов с событием берет на себя среда разработки. В модели представлены следующие обработчики событий:
  - *локализация* — обработчик, который следит за событием переключения языка интерфейса;
  - *контроль ввода-вывода* направлен на проверку (валидацию) входного значения перед выполнением преобразования;
  - *переключение типа и формата объекта* — обработчики событий, отслеживающие установку модификатора формата: тип числа (вещественное, дробное, интервальное) и способ кодирования (бинарное, постбинарное);
  - *перехват и обработка исключений* — комплекс перехватчиков исключений, возвращающих как наименование вызванного исключения, так и вызвавшую его функцию;
  - *визуализация битового представления числа* — обработчик, который вызывается после корректного завершения преобразования числа, в результате чего происходит заполнение полей на вкладках форматов (к.3, к.4, рис. 1);

– *отображение времени и статуса вычислений* — обработчик, служащий для расчета времени выполнения определенных частей алгоритма.

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 [25]). При этом для реализации такой арифметики используется числовой тип BigInteger пространства имен System.Numerics [26], являющийся сложным целочисленным типом, поддерживающим произвольное число больших целых чисел (не имеет нижней или верхней границы и может содержать любое целочисленное значение). Ядро программы включает в себя следующие блоки:

- *формат и тип числа* — набор переменных, содержащих в себе необходимую информацию об обрабатываемом числе (знак, порядок, мантисса, вид округления, формат и тип).
- *числа 32, 64, 128 и 256 бит* — экземпляры специального класса (FCCore.Number), которые хранят информацию о представлении числа для каждого формата: статус обработки числа; битовые поля; вид числа; смещение порядка; извлеченное точное значение; погрешность представления.
- *строковые преобразователи* — набор функций, которые работают со строковым представлением числа, проверяя и разделяя его на битовые поля мантиссы и порядка, подготавливая к арифметическим операциям и осуществляя

взаимообратный перевод из 2-чной в 16-ричную систему счисления.

- *арифметическое устройство* — блок функций оперирующих строковыми и целочисленными данными. При этом в строках содержатся записи входных и выходных числовых данных блока, а с целочисленными переменными (тип данных System.Numerics.BigInteger) производятся все арифметические операции над целой и дробной частью числа. Эти части обрабатываются по отдельности с последующим сведением результатов как к двоичным полям мантиссы, порядка и знака, так и к точному значению, извлеченному из сформированных полей постбинарного формата.
- *преобразователь чисел* — группа функций осуществляющих преобразование десятичного вещественного числа в двоичную дробь, а также приведение числа (двоичного и десятичного) к нормализованному виду.

На рис. 3.43 приведена диаграмма деятельности (activity diagram) в нотации языка UML [27], призванная детализировать особенности алгоритмической и логической реализации выполняемых программой операций.

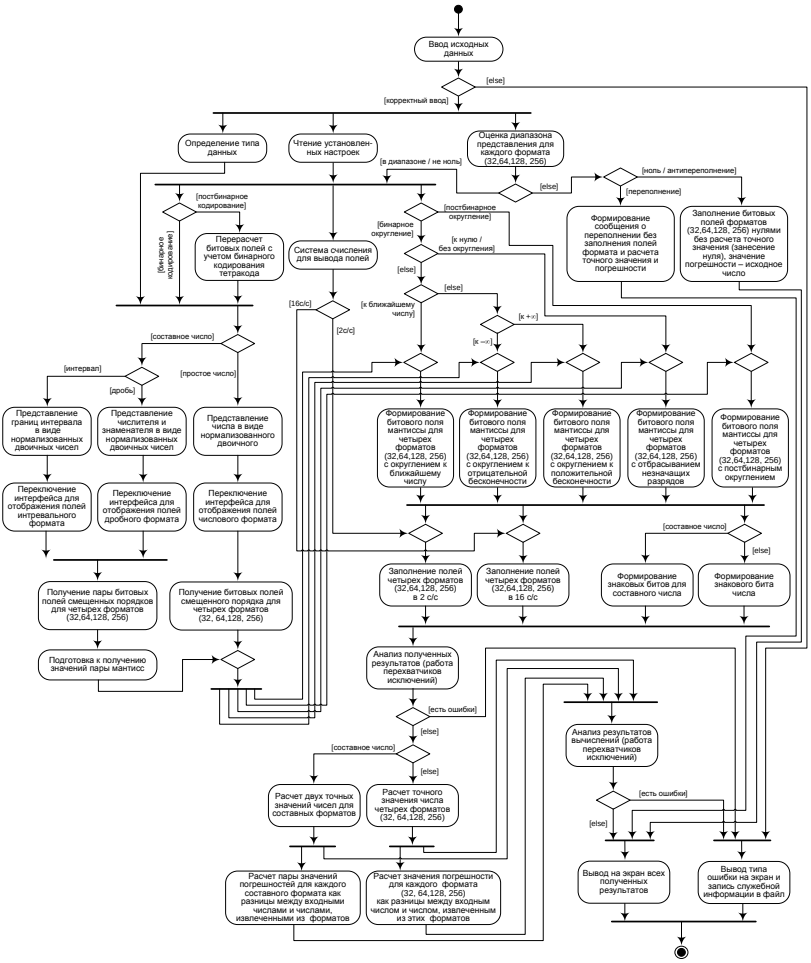
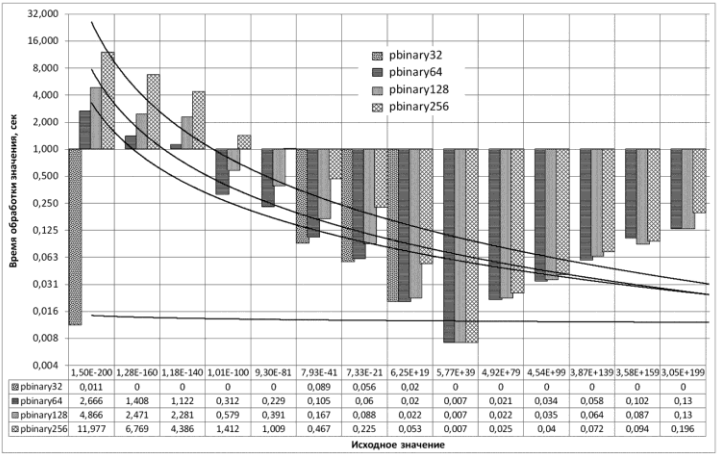


Рисунок 3.43 — Диаграмма деятельности программы PFC

На рис. 3.44 приведена гистограмма общего времени обработки программой PFC значений из диапазона 1,5E±200 (с логарифмическим шагом) и преобразования их в постбинарные форматы pb32 ÷ pb256. Нулевые значения

времени для формата pbinary32 показывают, что исходное число не вошло в область представления этого формата, и дальнейшая обработка не производилась. Общее время для каждого формата состоит из суммы затраченного времени на формирование битовых полей порядка, мантиссы и знака, извлечение точного значения из этих полей и расчета погрешности представления исходного числа. Измерения выполнены с точностью до миллисекунды.



	1,50E-200	1,28E-160	1,18E-140	1,01E-100	9,30E-81	7,93E-41	7,33E-21	6,25E+19	5,77E+39	4,92E+79	4,54E+99	3,87E+139	3,58E+159	3,05E+199
pbinary32	0,011	0	0	0	0	0,089	0,056	0,02	0	0	0	0	0	0
pbinary64	2,666	1,408	1,122	0,312	0,229	0,105	0,06	0,02	0,007	0,021	0,034	0,058	0,102	0,13
pbinary128	4,866	2,471	2,281	0,579	0,391	0,167	0,088	0,022	0,007	0,022	0,035	0,064	0,087	0,13
pbinary256	11,977	6,769	4,386	1,412	1,009	0,467	0,225	0,053	0,007	0,025	0,04	0,072	0,094	0,196

Рисунок 3.44 — Гистограмма временных затрат на обработку данных для постбинарных форматов и графики аппроксимирующих функций

Для нахождения причинно-следственной связи между входными данными и временем их обработки были получены коэффициенты логарифмической и степенной аппроксимирующих функций (кривые на рис. 3.44), численный вид которых представлен в табл. 3.13. Логарифмическая аппроксимирующая функция применена только для формата pbinary32 из-за наличия нулевых значений аргумента, при которых невозможно применить

степенную функцию. В табл. 3.13 также приведены значения величины R^2, которые характеризуют достоверность аппроксимации: чем ближе R^2 к единице, тем полученная функция точнее аппроксимирует исследуемый процесс.

Таблица 3.13 — Аппроксимирующие функции и величина достоверности аппроксимации R^2 для исследуемых форматов

Формат	*Аппроксимация*	*Аппроксимирующая функция f(x)*	R^2
pb32	Логарифмическая	$-0{,}0009 \ln(x) + 0{,}0141$	0,206
pb64	Степенная	$3{,}2991\, x^{-1{,}887}$	0,7742
pb128	Степенная	$7{,}8003\, x^{-2{,}157}$	0,821
pb256	Степенная	$26{,}207\, x^{-2{,}508}$	0,8476

На основании полученных результатов 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:

```
1,00100101111011101101100011111111110110011101100e-664;
```

- для pbinary128:

  1,001001011110111011011000111111111011001110011100000100010111101100011110010010011111101001100110000011010e-664;

- для pbinary256:

  1,0010010111101110110110001111111110110011100111000001000101111011000111100100100111111010011001100000110011000100111001111111010110111001101011111111101010010101101011111110010011011010100010101100100010001101111100100111e-664

в соответствующие десятичные дроби:

- для pbinary64:

  1,4999999999999996830668416126683557766694463834122956273169886045355187178442727891994362741918888334315343923308207170967744346305706583068382429364958665499574673400370351047036375195939993158407568317077135113315489940591358231084405519120675866580119284581254022802640715521322226012093728056717130968012576813815380685440020326456171347353241477997512793597703292517866507460278193283416442987674273509551655239607672702679192194789177732503845190983985679946012678151467945042440987890586256980895996093 75e-200;

- для pbinary128:

  1,500000000000000000000000000000029736925385879603512981657336393886180844159622543222587667348491274409135991825637900579782014114689514001626100114350697931034298947399662706411139733715495495717201445294803323000740894900404976411385058911077819153397307301220041872674712966875057862554268072845031235122527745649807629541176195615183891700838589512934935300732062585961874920218562328773990028956077894356832191218416630769452484341896155337332103079976056754911538113122610436374545394080319648658670535054051412306221091087776731143321740091778337955474853515625e-200;

- для pbinary256:

  1,499999999999999999999999999999999999999999999999999999999999999998219670476521811139347520283036754598568712319942345968144746684806694886351128204388

```
8954023549664869965327127303020313596213436760436305857080620690580672100607213236194344171866467422297439560461273736196341507418363558985582382478249298925161067583559513607854125325581150168131634044310336491291374096096618689331619858464991533829036717321144420911536844767704099632875001045926275065877616845818337228719704425692510666990330049456470295601275873300665895365898081504965465734685505291741738399750833713348056839472052724443295165741992065325454077336677868196276568603202139495778055788832716643810272216796875e-200.
```

Данный пример демонстрирует вычислительные возможности программы. Причем для получения значения погрешности представления чисел организована операция вычитания по модулю между рассчитанным «длинным» и исходным числами для каждого формата.

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

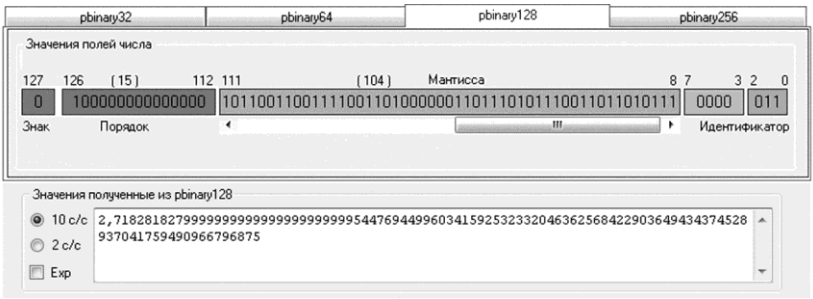
- числа $e \approx 2{,}718281828$;
- рациональной дроби $e \approx \frac{271801}{99990}$;
- интервала $e \in [2{,}718281828;\ 2{,}718281829]$.

На рис. 3.45 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 табл. 3.14 сведены погрешности представления исходных данных для каждого формата.

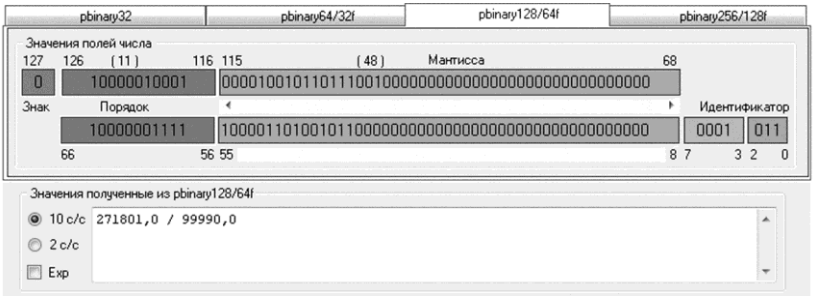
Таблица 3.14 — Погрешность представления константы *e* в числовых дробных и интервальных форматах

Разрядность	*Число*	*Дробь*	*Интервал*
32	8,208935547e–7	–*	–
64	2,064982255e–14	0,0 / 0,0	8,208935547e–7; 8,308935547e–7
128	4,552305500e–31		2,064982255e–14; 1,4049910533e–14
256	2,375392996e–66		4,552305500e–31; 8,608605556e–32

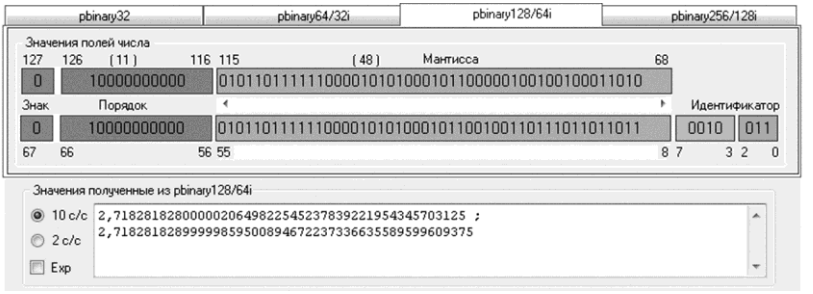
* — работа с дробями и интервалами не предусмотрена (см. табл. 3.12 и Приложение.)



а)



б)



в)

Рисунок 3.45 — Представление константы e в 128-разрядных постбинарных форматах в виде десятичной (*а*) и рациональной (*б*) дробей, а также в виде вещественного интервала (*в*)

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.

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.

На рис. 3.46 продемонстрирована работа программы PFC: 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.

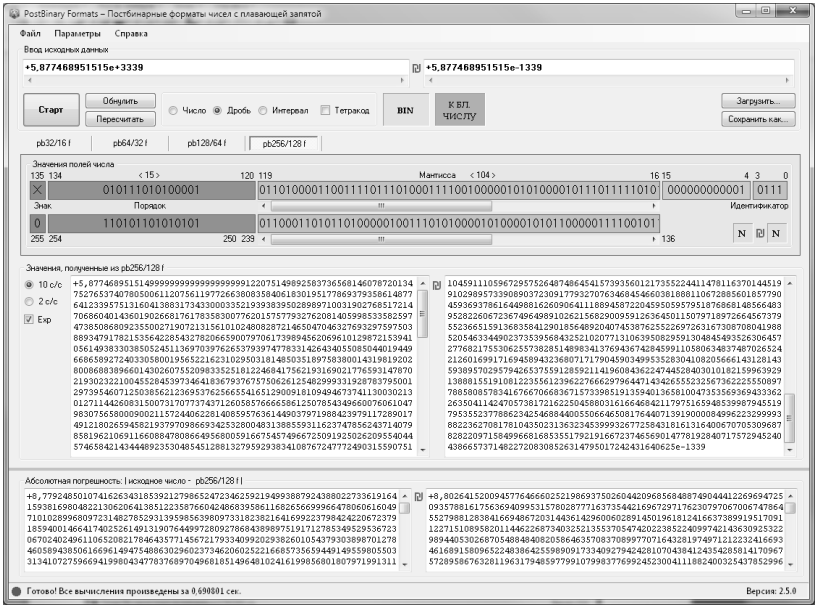


Рисунок 3.46 — Преобразователь PFC (обновленная версия) в режиме точного отображения десятичных чисел, извлеченных из полей формата pb256/128f

3.7. Выводы

Полученные в п. 3.1 компьютерные компоненты рассмотренных в п. 2.4 операций тетралогики могут выступать в качестве элементного базиса для реализации специализированных компонентов, способных реализовывать базовые операции тетраарифметики, т. е.

производить тетравычисления на базе современных компьютерных систем.

Реализация тетраарифметики и тетравычислений в современных бинарных компьютерах является **весомым аргументом в пользу постепенного перехода к постбинарному компьютингу уже на нынешнем этапе развития компьютерных технологий** и делает возможным создание вычислительных моделей, способных осуществлять достоверные вычисления и представлять в вычисляемом виде множество противоречий окружающего мира.

Использование рассмотренных в п. 3.2 двоичных сумматоров для получения тетрасуммы делает возможным использовать известные в цифровой схемотехнике подходы в организации многоразрядных сумматоров и методы ускорения операции суммирования. Поскольку арифметико-логическое устройство (АЛУ) центрального процессора обязательно содержит в своем составе сумматоры, то синтез постбинарных суммирующих компонентов является ключевым для последующего проектирования АЛУ постбинарного сопроцессора [2, 28, 29].

В п. 3.4 рассмотрен способ постбинарного кодирования интервалов путем сведения тетракода к значениям интервальных границ. Предложены способы интервального оценивания (оценки предельных колебаний) границ интервала с использованием функций алгебры тетралогики. Представлена программная реализация рассматриваемой задачи — программа декодирования тетракода, которая позволяет выполнять:

- преобразование тетракода в набор целочисленных значений;
- преобразование тетракода в набор вещественных чисел;
- декодирование тетракода в границы интервалов с проведением интервального оценивания «сверху» и «снизу».

Представленный программный продукт является в своем роде уникальным и может служить в учебных целях для наглядной иллюстрации представления тетракодов в виде числовых (точечных) и интервальных (множественных) значений.

В п. 3.5 рассмотрена точность представления чисел с плавающей запятой и показаны формулы для расчета абсолютной и относительной погрешностей для бинарных и постбинарных форматов. Получены формулы для расчета погрешностей всех форматов и выявлена закономерность между значениями погрешности постбинарного и бинарного форматов одного класса точности. Рассмотрены стандартные способы округления чисел с плавающей запятой и предложен новый способ округления чисел в постбинарных форматах. Следует отметить, что только с помощью механизма постбинарного округления становится возможным эффективно использовать тетракоды для кодирования множества (в общем случае, интервала) действительных чисел одним тетракодовым значением.

В п. 3.6 рассмотрена программа-преобразователь данных в постбинарные форматы с плавающей запятой. Программа Postbinary Format Converter позволяет значительно снизить трудозатраты, связанные с преобразованием операндов и оценить погрешность их представления в постбинарных форматах чисел с

плавающей запятой. Кроме того, программа снабжена обширной справочной информацией о представленных форматах: характеристикой, значениями минимальных и максимальных числовых диапазонов, относительной точности десятичных цифр и значениями исключительных чисел.

Таким образом, на сегодняшний день авторами реализован весь комплекс простейших постбинарных арифметических операций. Кроме того, продемонстрирована эффективность использования постбинарных форматов с плавающей запятой, реализованы программные приложения для тетроарифметики и продолжаются работы по программной и аппаратной реализации постбинарных вычислений.

Заключение

Несмотря на публикацию авторами с начала 1990-х годов около 40-ка докладов и статей, а также 2-х монографий по постбинарному компьютингу (в 2011 и 2012 гг.), следует констатировать, что исследования в данном направлении находятся пока еще в начальной стадии. И данная монография — это всего лишь очередной шаг на длинном пути к реальному и масштабному использованию всех преимуществ постбинарного кодологического базиса.

Цель данной работы заключалась в привлечении внимания специалистов и широкого круга пользователей к проблематике постбинарного компьютинга, а также — в анализе и обобщении всего того, что удалось уже сделать в данной области за первые годы и десятилетия исследований.

Сделать, к сожалению, удалось пока не так много, так как все исследования и разработки базировались в основном на энтузиазме авторов, а также – на таланте и трудолюбии привлекаемых ими к исследованиям и разработкам студентов и аспирантов факультета компьютерных наук и технологий Донецкого национального технического университета. Тем не менее, объем сделанного уже вполне достаточен для того, чтобы однозначно утверждать об актуальности и перспективности данного направления исследований, а также — вплотную переходить к постановке и решению реальных задач по практической реализации всех преимуществ постбинарного компьютинга в реальных системах.

Путь впереди и долгий, и сложный, и неоднозначный. Но авторы выражают надежду, что могут заразить своим энтузиазмом и других, возможно, более талантливых и настойчивых, исследователей и разработчиков в области компьютерной инженерии, в первую очередь, молодых, у которых впереди будет достаточно времени, чтобы в полной мере насладиться результатами своего труда и увидеть те результаты, которые сегодня кажутся пока еще слишком труднодостижимыми и малореальными.

В приложении к монографии приведен детально проработанный набор постбинарных форматов представления данных, который может уже сейчас служить основой для разработки реальных постбинарных алгоритмов, приложений и систем. Многие другие принципиальные вопросы, связанные с постбинарным представлением и обработкой данных также успешно решены. Но требуются новые идеи и свежий критический взгляд на возможности эффективного применения постбинарного компьютинга в реальных приложениях.

Речь при этом может идти не только о сугубо компьютерной инженерии. Существенный вклад в успешное продвижение к постбинарному будущему могут внести профессионалы в самых разных областях, начиная от программной инженерии и заканчивая математикой и философией. Анализ растянувшегося на многие столетия перехода от монокодовой эпохи к цифровой бинарной, показал сколь много самых разных ученых, изобретателей и предпринимателей внесли свой решающий вклад в этот очередной качественный скачок в интеллектуальной истории человечества.

Главное помнить, что процесс интеллектуального восхождения цивилизации и совершенствования ее

интеллектуального инструментария является главным залогом ее развития. И каждый, кто может, должен внести посильный вклад в это благородное дело.

Сделать будущие возможности реальностью уже сегодня – это, пожалуй, главное устремление, согласно которому многочисленные исследователи, изобретатели, инноваторы и предприниматели обеспечили невероятный прогресс компьютерных технологий во второй половине XX века и в начале нового 3-го тысячелетия. Благодаря этому, как представляется, и постбинарный компьютинг может стать повседневной реальностью уже в обозримом будущем.

СПИСОК ЛИТЕРАТУРЫ

К главе 1

1. Аноприенко А.Я. Повышение производительности систем генерации изображений: структуры и алгоритмы на уровне регенерационной памяти: автореферат диссертации на соискание ученой степени кандидата технических наук. — Киев, 1987. — 20 с.
2. Моисеев Н.Н. Алгоритмы развития (серия «Академические чтения»). — М: Наука, 1987. — 304 с.
3. Моисеев Н.Н. Математические задачи системного анализа. — М.: Наука, 1981. — 488 с.
4. Моисеев Н.Н., Александров В.В., Тарко А.М. Человек и биосфера: Опыт системного анализа и эксперименты с моделями. – М.: Наука, 1985. — 271 с.
5. Крик Ф., Ниренберг М. Генетический код // «Успехи физических наук», 1964 г., январь. С. 133–160.
6. Белнап Н. Как нужно рассуждать компьютеру // В книге «Логика вопросов и ответов». — М.: «Прогресс», 1981. — 288 с.
7. Бахтияров К.И. Как может рассуждать компьютер в духе булевой многозначности // «Логические исследования». Выпуск 17. — М.-СПб: ЦГИ, 2011. — 20 с.

8. Аноприенко А.Я. Обобщенный кодо-логический базис в вычислительном моделировании и представлении знаний: эволюция идеи и перспективы развития // Научные труды Донецкого национального технического университета. Серия «Информатика, кибернетика и вычислительная техника» (ИКВТ-2005) выпуск 93: — Донецк: ДонНТУ, 2005. С. 289–316.

9. Аноприенко А.Я. Постбинарный компьютинг: развитие представлений о многомерном логическом пространстве и связь с ноокомпьютингом // Информатика и компьютерные технологии / Сборник трудов IX международной научно-технической конференции 4-6 ноября 2013 г., Донецк, ДонНТУ. — 2013. С. 488–509.

10. Аноприенко А.Я., Кухтин А.А. О некоторых возможностях расширения логического базиса информатики. // Тезисы докладов международной научно-практической конференции «Информатизация в условиях перехода к рынку», Киев, 5-6 ноября 1992. С. 30–32.

11. Аноприенко А.Я. Тетралогика и тетракоды. // Сборник трудов факультета вычислительной техники и информатики. Вып.1. — Донецк: ДонГТУ. — 1996. С. 32–43.

12. Анопрієнко О., Кривошеєв С. Тетракоди: новий метод кодування сигналів і зображень // Праці Всеукраїнської міжнародної конференції “Оброблення сигналів і зображень та розпізнавання образів (УкрОБРАЗ’96)". — Київ. — 1996. С. 15–17.

13. Аноприенко А. Я., Кривошеев С. В., Приходько Т. А. Тетракоды в кодировании и распознавании образов // Сборник научных трудов ДонГТУ. Серия «Информатика, кибернетика и вычислительная

техника». Выпуск 1 (ИКВТ-97). — Донецк: ДонГТУ. — 1997. — С. 99–104.

14. Аноприенко А.Я. Ноографика и ноомоделирование // Материалы четвертой международной научно-технической конференции «Моделирование и компьютерная графика» 5-8 октября 2011 года, Донецк, ДонНТУ, 2011. С. 321-324.

15. Аноприенко А.Я. Ноокомпьютинг и будущее информационно-компьютерной инфраструктуры // Международный научный конгресс по развитию информационно-коммуникационных технологий и развитию информационного общества в Украине, г. Киев, 17–18 ноября 2011 года. Тезисы докладов. С. 12–13.

16. Аноприенко А.Я. Ноокомпьютинг // Материалы VI международной научно-технической конференции «Информатика и компьютерные технологии» — 22–23 ноября 2011 года. Т. 1. Донецк, ДонНТУ. — 2011. С. 10–23.

17. Аноприенко А.Я. Программная инженерия и обобщенный закон Мура // Первая международная научно-практическая конференция «Программная инженерия: методы и технологии разработки информационно-вычислительных систем (ПИИВС-2016). Донецк, 16-17 ноября 2016 г. Сборник научных трудов. – Донецк: ГОУ ВПО «Донецкий национальный технический университет», 2016. — С. 41–47.

18. Аноприенко А.Я. Закономерности развития компьютерных технологий и обобщенный закон Мура // Вестник Донецкого национального технического университета, №2 (2), 2016. С. 3–17.

19. Аноприенко А.Я. Пятая волна индустриализации и третья промышленная революция // Вестник

Донецкого национального технического университета, № 1 (1), 2016. С. 3–12.

20. Аноприенко А.Я. Системодинамика техносферы: как измерить технический прогресс // Системный анализ и информационные технологии в науках о природе и обществе, 2015. № 1(8)-2(9). С. 47–58.

21. Аноприенко А.Я. Периодическая система развития компьютерных систем и перспективы нанокомпьютеризации // Инновационные перспективы Донбасса: Материалы международной научно-практической конференции. Донецк, 20-22 мая 2015 г. Том 5. Компьютерные науки и технологии. — Донецк: Донецкий национальный технический университет, 2015. С. 5–13.

22. Аноприенко А.Я. Системодинамика ноотехносферы: основные закономерности // «Системный анализ в науках о природе и обществе». — Донецк: ДонНТУ, 2014, № 1(6)–2(7). С. 11–29.

23. Аноприенко А.Я., Варзар Р.Л., Иваница С.В. Закономерности развития аналого-цифровых преобразователей и перспективы использования постбинарного кодирования // Научные труды Донецкого национального технического университета. Серия: «Информатика, кибернетика и вычислительная техника» (ИКВТ-2014). Выпуск 1 (19). — Донецк: ДонНТУ, 2014. С. 5–10.

24. Аноприенко А.Я. Четыре концепции будущего: «Зеленый рост», «Индустрия 4.0», нооинфраструктура и космоантропная перспектива // Донбасс-2020: Материалы VII научно-практической конференции. Донецк, 20–23 мая 2014

г. – Донецк, Донецкий национальный технический университет, 2014. С. 6–11.

25. Аноприенко А.Я. Система закономерностей развития средств и методов компьютинга // Материалы V всеукраинской научно-технической конференции «Информационные управляющие системы и компьютерный мониторинг (ИУС и КМ 2014)» — 22–23 апреля 2014 г., Донецк, ДонНТУ, 2014. В 2-х томах. Т. 1. С. 11–23.

26. Аноприенко А.Я. Основные закономерности эволюции компьютерных систем и сетей // Научные труды Донецкого национального технического университета. Серия «Проблемы моделирования и автоматизации проектирования» (МАП-2013). Выпуск № 1 (12) – 2 (13): Донецк: ДонНТУ, — 2013. С. 10–32.

27. Аноприенко А.Я. Модели эволюции компьютерных систем и средств компьютерного моделирования // Материалы пятой международной научно-технической конференции «Моделирование и компьютерная графика» 24-27 сентября 2013 года, Донецк, ДонНТУ, 2013. С. 403–423.

28. Щербаков А.С., Аноприенко А.Я. Основные особенности и перспективы развития моделирующих сред // Материалы IV всеукраинской научно-технической конференции «Информационные управляющие системы и компьютерный мониторинг (ИУС и КМ 2013)» – 24-25 апреля 2013 г., Донецк, ДонНТУ, 2013. В 2-х томах. Т. 1. С. 257–261.

29. Аноприенко А.Я., Святный В.А. Высокопроизводительные информационно-моделирующие среды для исследования, разработки и сопровождения сложных динамических систем //

Научные труды Донецкого государственного технического университета. Выпуск 29. Серия "Проблемы моделирования и автоматизации проектирования динамических систем" — Севастополь: «Вебер». — 2001. — С. 346–367.

30. Аноприенко А.Я., Святный В.А. Универсальные моделирующие среды // Сборник трудов факультета вычислительной техники и информатики. Вып.1. - Донецк:
ДонГТУ. - 1996. С. 8-23.

31. Святный В.А., Цайтц М., Аноприенко А.Я. Реализация системы моделирования динамических процессов на параллельной ЭВМ в среде сетевого графического интерфейса // Вопросы радиоэлектроники, серия «ЭВТ», вып. 2. — 1991. С. 85–94.

32. Аноприенко А.Я., Варзар Р.Л. Разработка прототипа суперсенсорного компьютера: особенности реализации и визуализации результатов измерений // Материалы пятой международной научно-технической конференции «Моделирование и компьютерная графика» 24-27 сентября 2013 года, Донецк, ДонНТУ, 2013. С. 218–229.

33. Варзар Р.Л., Аноприенко А.Я. Аппаратная архитектура сенсорного модуля суперсенсорного компьютера и его параметры // Материалы IV всеукраинской научно-технической конференции «Информационные управляющие системы и компьютерный мониторинг (ИУС и КМ 2013)» — 24-25 апреля 2013 г., Донецк, ДонНТУ, 2013. В 2-х томах. Т. 1. С. 720–728.

34. Варзар Р.Л., Аноприенко А.Я. Суперсенсорный компьютер для измерения и анализа параметров окружающей среды // Информатика и

компьютерные технологии / Сборник трудов VIII международной научно-технической конференции 18-19 сентября 2012 г., Донецк, ДонНТУ. – 2012. В 2-х томах. Т. 2. С. 156–161.

35. Varzar R., Anopriyenko A. Supersensory computer: idea and architecture // Proceedings of the XIII International Young Scientists' Conference on Applied Physics, 12-15 June 2013, Taras Shevchenko national university of Kiev. 2013. P. 230-231.

36. Varzar R., Anopriyenko A. Supersensory computers for measurement and analysis of biologically dangerous factors of environment / Theoretical and Applied Aspects of Cybernetics - 2012 / Proceedings of the 2nd International Scientific Conference of Students and Young Scientists - Kyiv: "Bukrek" , Taras Shevchenko national university of Kiev - 2012, p. 186-191.

37. Аноприенко А.Я. Будущее компьютерных технологий в контексте технической и кодологической эволюции // Вестник Инженерной Академии Украины. Теоретический и научно-практический журнал Инженерной Академии Украины. Выпуск 3-4, 2011. С. 108–113.

38. Аноприенко А.Я. Вызовы времени и постбинарный компьютинг // Материалы VI международной научно-технической конференции «Информатика и компьютерные технологии» – 23–25 ноября 2010 г. Т. 1. Донецк, ДонНТУ. – 2010. С. 13–31.

39. Аноприенко А.Я., Коноплева А.П. Опыт применения гиперкодов в моделировании клеточных автоматов // Научные труды Донецкого национального технического уни-верситета. Серия «Проблемы моделирования и автоматизации проектирования динамических систем» (МАП-

2007). Выпуск 6 (127): Донецк: ДонНТУ, 2007. С. 220-227.

40. Аноприенко А.Я., Коноплева А.П. Развитие идеи применения гиперкодов в моделировании клеточных автоматов // Научные труды Донецкого национального технического университета. Серия: «Информатика, кибернетика и вычислительная техника» (ИКВТ-2008). Выпуск 9 (132): – Донецк: ДонНТУ, 2008. С. 115–118.

41. Аноприенко А.Я., Коноплева А.П., Василенко А.Ю. Оценка производительности при моделировании постбинарных клеточных автоматов и способы ее повышения // Научные труды Донецкого национального технического университета. Выпуск 147 (16). Серия «Вычислительная техника и автоматизация». – Донецк, ДонНТУ, 2009 С. 96–104.

42. Аноприенко А.Я., Коноплева А.П. Моделирование постбинарных клеточных автоматов // «Моделирование и информационные технологии». Сборник научных трудов. Специальный выпуск по материалам международной научной конференции «Моделирование-2010» (13-14 мая 2010 года). – Киев, НАН Украины, Институт проблем моделирования в энергетике им. Г.Е. Пухова, 2010. С. 162-170.

43. Аноприенко А.Я., Коноплева А.П. Управляемые и неуправляемые постбинарные клеточные автоматы // Материалы V Всеукраинской научно-практической конференции «Современные тенденции развития информационных технологий в науке, образовании и экономике», Луганск, 7–8 апреля 2011 г., том 1. С.19–21.

44. Аноприенко А.Я., Коноплёва А.П., Плотников Д.Ю., Малёваный Е.Ф. Применение клеточных автоматов для моделирования динамических процессов: опыт ДонНТУ // Материалы четвертой международной научно-технической конференции «Моделирование и компьютерная графика» 5-8 октября 2011 года, Донецк, ДонНТУ, 2011. С. 271–278.

45. Аноприенко А.Я., Плотников Д.Ю., Малёваный Е.Ф., Коноплёва А.П. Варианты эффективной реализации постбинарных клеточных автоматов // Материалы VI международной научно-технической конференции «Информатика и компьютерные технологии» – 22–23 ноября 2011 г. Т. 1. Донецк, ДонНТУ. – 2011. С. 238–241.

46. Аноприенко А.Я., Федоров Е.Е., Иваница С.В., Альрабаба Хамза, Особенности аппаратной реализации обобщенного клеточного тетраавтомата // Технологический аудит и резервы производства, № 1/3 (21), 2015. С. 68–74.

47. Аноприенко А.Я., Иваница С.В. Постбинарный компьютинг и интервальные вычисления в контексте кодо-логической эволюции. — Донецк, ДонНТУ, УНИТЕХ, 2011. — 244 с.

48. Аноприенко А.Я., Иваница С. В. Тетралогика, тетравычисления и ноокомпьютинг. Исследования 2010–2012. — Донецк: ДонНТУ, Технопарк ДонНТУ УНИТЕХ, 2012. — 308 с.

49. Аноприенко А.Я. Расширенный кодо-логический базис компьютерного моделирования / В кн. «Информатика, кибернетика и вычислительная техника (ИКВТ-97). Сборник научных трудов ДонГТУ». Выпуск 1. Донецк, ДонГТУ, 1997, с. 59–64.

50. Аноприенко А. Я. Эволюция алгоритмического базиса вычислительного моделирования и сложность реального мира // Научные труды Донецкого национального технического университета. Выпуск 52. Серия «Проблемы моделирования и автоматизации проектирования динамических систем» (МАП-2002): Донецк: ДонНТУ, 2002. — С. 6–9.

51. Маковельский А.О. История логики. — М.: Наука, 1967. — 502 с.

52. Anoprienko A., Svjatnyi V., Reuter A. Extended logical and numerical basis for computer simulation / “Short Papers Proceedings of the 1997 European Simulation Multiconference ESM'97. Istanbul, June 1–4, 1997" — Istanbul, SCS, 1997, p. 23-26.

53. Anoprienko A. Interpretation of some artifacts as special simulation tools and environments / “Short Papers Proceedings of the 1997 European Simulation Multiconference ESM'97. Istanbul, June 1–4, 1997" — Istanbul, SCS, 1997, p. 23–26.

54. Anoprienko A. Tetralogic and tetracodes: an effective method for information coding // 15th IMACS World Congress on Scientific Computation, Modeling and Applied Mathematics. Berlin, August 24-29, 1997. Vol. 4. Artificial Intelligence and Computer Science. – Berlin: Wissenschaft und Technik Verlag. — 1997. — P. 751–754.

55. Аноприенко А.Я. От вычислений к пониманию: когнитивное компьютерное моделирование и его практическое применение на примере решения проблемы Фестского диска / В кн. «Информатика, кибернетика и вычислительная техника (ИКВТ-99). Сборник научных трудов ДонГТУ». Выпуск 6. Донецк, ДонГТУ, 1999, с. 36–47.

56. Аноприенко А.Я. Восхождение интеллекта: эволюция монокодовых вычислительных моделей // Научные труды Донецкого государственного технического университета. Выпуск 15. Серия «Информатика, кибернетика и вычислительная техника» (ИКВТ-2000). — Донецк: ДонГТУ. — 2000. — С. 87–107.

57. Anoprienko A. Archaeosimulation: new sight on ancient society and lessons for computer era / Problems of Simulation and Computer Aided Design of Dynamic Systems. Scientific Papers of Donetsk State Technical University. Vol. 29. – Sevastopol: Weber, 2001. P. 320-326.

58. Anoprijenko A. The early history of simulation in Europe: scale planetariums and astromorphic models // EUROSIM 2004: 5th EUROSIM Congress on Modeling and Simulation. 06–10 September 2004. ESIEE Paris, Marne la Vallée, France. Book of abstracts. S. 146–147.

59. Аноприенко А. Я. Модельная и компьютерная поддержка принятия решений в ситуации когнитивного конфликта: рассмотрение на примере сравнительного анализа гипотез о локализации Атлантиды Платона // Научные труды Донецкого национального технического университета. Выпуск 52. Серия «Проблемы моделирования и автоматизации проектирования динамических систем» (МАП-2002): Донецк: ДонНТУ, 2002. — С. 177-243.

60. Batens D. Inconsistency-adaptive logics and the foundation of non-monotonic logic // Logique et Analyse. 1994. N 145. P. 57–94.

61. История логики. – Мн.: Новое знание, 2001. — 170 с.

62. Аноприенко А.Я. Археомоделирование: Модели и инструменты докомпьютерной эпохи. – Донецк: УНИТЕХ, 2007. – 318 с.

63. Васильев Н.А. О частных суждениях, о треугольнике противоположностей, о законе исключенного четвертого // Ученые записки императорского Казанского университета. — Казань: Типо-литография Университета, 1910. Октябрь. С. 1–47.

64. Бажанов В.А. Н. А. Васильев и его воображаемая логика. Воскрешение одной забытой идеи. — М., 2009. – 240 с.

65. Brouwer L.E.J. De onbetrouwbaarheid der logische principes // Tijdschrift voor Wijsbegeerte. 1908. Vol. 2. P. 152–158.

66. Брауэр Л.Э.Я. Недостоверность принципов логики // «Логические исследования», 2016. Т. 22. № 1. С. 171–176.

67. Лукасевич Я. О принципе противоречия у Аристотеля. Критическое исследование / Пер. с польского издания 1910 г. – М. – СПб.: ЦГИ, 2012. – 256 с.

68. Lukasiewicz J. O logice tr´ojwarto´sciowej // Ruch Filozoficzny, 1920. N. 5. S. 169–171.

69. Карпенко А.С. Развитие многозначной логики. – М.: Издательство ЛКИ, 2010. – 448 с.

70. Аноприенко А.Я. Постбинарный компьютинг и моделирование сложных систем в контексте кодологической эволюции // Доклад на международной научной конференции «Моделирование-2010» (13-14 мая 2010 года). – Киев, НАН Украины, Институт проблем моделирования в энергетике им. Г.Е. Пухова, 2010.

71. Friedlein G. Boetii de institutione arithmetica. – Lipsiae, 1876, p.4–5.

72. Wardley P., White P. The Arethmeticke project: a collaborative research study of the diffusion of hindu-arabic numerals // Family & Community History, Vol.6/1, May 2003. P. 5-17.

73. Апокин И.А., Майстров Е.М. Развитие вычислительных машин. – М.: Наука, 1974. – 399 с.

74. Ларичев В.Е. Мудрость змеи: Первобытный человек, Луна и Солнце. – Новосибирск: Наука, 1989. – 272 с.

75. Омельченко В.В. О законе циклического развития мира, действительности, бытия в знаковых образах Мальтинской пластины из бивня мамонта времен палеолита // Вестник археологии, антропологии и этнографии. 2016. № 2 (33). С. 16–29.

76. Лейбниц Г. В. Сочинения в 4-х томах, т. 3. — М.: Мысль, 1984. – 734 с.

К главе 2

1. Аноприенко А. Я. Археомоделирование: модели и инструменты докомпьютерной эпохи. — Донецк: УНИТЕХ, 2007. — 318 с.
2. Белнап Н. Как нужно рассуждать компьютеру / Н. Белнап, Т. Стил // В кн. Логика вопросов и ответов. — М.: «Прогресс», 1981. — 288 с.
3. Аноприенко А. Я., Иваница С. В. Постбинарный компьютинг и интервальные вычисления в контексте кодо-логической эволюции. / А. Я. Аноприенко, С. В. Иваница — Донецк, ДонНТУ, УНИТЕХ, 2011. — 248 с.
4. Аноприенко А. Я. О некоторых возможностях расширения логического базиса информатики. / А. Я. Аноприенко, А. А. Кухтин // В кн. «Тези доповідей міжнародної науково-практичної конференції "Інформатизація в умовах переходу до ринку"», Київ, 5–6 листопада 1992 р., — С. 30–32.
5. Аверкин А. Н. Толковый словарь по искусственному интеллекту / А. Н. Аверкин, М. Г. Гаазе-Рапопорт, Д. А. Поспелов — М.: Радио и связь, 1992. — 256 с.
6. Аноприенко А. Я. Обобщенный кодо-логический базис в вычислительном моделировании и представлении знаний: эволюция идеи и перспективы развития // Научные труды Донецкого национального технического университета. Серия «Информатика, кибернетика и вычислительная техника» (ИКВТ-2005), выпуск 93: — Донецк: ДонНТУ, 2005, С. 289–316.
7. Брусенцов Н. П. Трехзначная интерпретация силлогистики Аристотеля / Историко-

математические исследования. Вторая серия, выпуск 8 (43). — М.: «Янус-К», 2003. — С. 317–327.

8. Брусенцов Н. П. Интеллект и диалектическая триада / Н. П. Брусенцов // Искусственный интеллект — 2002. — № 2. — С.53–57.

9. История троичной логики. Материал из свободной русской энциклопедии «Традиция». —: http://traditio.ru/wiki/Троичная_логика.

10. Hayes N. T. Trits to Tetrits. — [Электронный ресурс] http://grouper.ieee.org/groups/1788/Position-Papers/Tetrits.pdf.

11. Информационная энтропия. Материал из Википедии — свободной энциклопедии. http://ru.wikipedia.org/wiki/Информацион-ная_энтропия.

12. Яблонский С. В. Введение в дискретную математику: учебное пособие для вузов. — 2-е изд., перераб. и доп. / С. В. Яблонский. — М.: Наука, 1986. — 384 с.

13. Ивин А. А. Логика. Учебное пособие; 2-е изд. / А. А. Ивин. — М.: Знание, 1998. — 228 с.

14. Марченков С. С. Замкнутые классы булевых функций / С. С. Марченков. — М.: ФизМатЛит, 2000. — 128 с.

15. Weisstein E. W. Hasse Diagram. From MathWorld — A Wolfram Web Resource. http://mathworld.wolfram.com/Has-seDiagram.html.

16. Колмогоров А. Н. Математическая логика, изд 3-е, стереотипное / А. Н. Колмогоров, А. Г. Драгалин — М.: КомКнига, 2006. — 256 с.

17. Аноприенко А. Я., Иваница С. В. Особенности постбинарного кодирования на примере

интервального представления результатов вычислений по формуле Бэйли-Боруэйна-Плаффа // Научные труды Донецкого национального технического университета. Серия: «Информатика, кибернетика и вычислительная техника» (ИКВТ-2010). Выпуск 11 (164). — Донецк: ДонНТУ, 2010, с. 19–23.

18. Иваница С. В., Аноприенко А. Я. Особенности реализации операций тетралогики // Научные труды Донецкого национального технического университета. Серия: «Информатика, кибернетика и вычислительная техника» (ИКВТ-2011). Выпуск 13 (185). — Донецк: ДонНТУ, 2011. С. 134–140.

19. Аноприенко А. Я., Иваница С. В. Особенности реализации постбинарных логических операций // Научно-теоретический журнал «Искусственный интеллект», № 2, 2011. С. 110–121.

20. Аноприенко А. Я. Иваница С. В., Бурлака Е. В. Программная реализация постбинарного кодирования интервалов / А. Я. Аноприенко, С. В. Иваница, Е. В. Бурлака // Научные труды Донецкого национального технического университета. Серия «Проблемы моделирования и автоматизации проектирования динамических систем» (МАП-2011). Выпуск 10 (197): Донецк: ДонНТУ, — 2011. С. 207–214.

21. 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 Международной научно-технической конференции (пос. Кацивели, АР Крым, 1–5 октября 2012 года). — Донецьк: ІПШІ «Наука і освіта», 2012. — С. 13–16.

22. Анопрієнко О. Я. Принцип роботи, структура і моделювання блоку перетворювача форматів у складі постбінарного співпроцесора / О. Я. Анопрієнко, С. В. Іваниця, С. В. Кулібаба // Міжнародний науково-технічний журнал «Інформаційні технології та комп'ютерна інженерія», № 1 (26). — Вінниця, 2013. — С. 59–65.

23. Шарый С. П. Конечномерный интервальный анализ. Институт вычислительных технологий СО РАН, изд. XYZ, 2013 — 605 с.

24. Bailey David H., Borwein Peter B., Plouffe Simon On the Rapid Computation of Various Polylogarithmic Constants // Mathematics of Computation. — 1997. — В. 218. Т. 66. — p. 903–913.

25. IEEE Standard for Floating-Point Arithmetic (Revision of IEEE Std 754-1985) / IEEE Computer Society — New York, NY 10016-5997, USA, 29 August 2008 — P. 70.

26. Добронец Б. С. Интервальная математика: Учеб. пособие. Краснояр. гос. ун-т. — Красноярск. 2004. — 216 с.

27. Аноприенко А. Я., Иваница С. В. Тетралогика, тетравычисления и ноокомпьютинг. Исследования 2010–2012 / А. Я. Аноприенко, С. В. Иваница — Донецк, ДонНТУ, УНИТЕХ, 2012. — 308 с.

28. Knuth D. The Art of Computer Programming. Volume 2: Seminumerical Algorithms. – Massachusetts: Addison-Wesley, 1997, — 762 p.

К главе 3

1. Аноприенко А.Я. Тетралогика и тетракоды. // Сборник трудов факультета вычислительной техники и информатики. Вып.1. – Донецк: ДонГТУ. – 1996. С. 32-43.

2. Аноприенко А. Я., Иваница С. В. Тетралогика, тетравычисления и ноокомпьютинг. Исследования 2010–2012. — Донецк, ДонНТУ, УНИТЕХ, 2012. — 308 с.

3. Баркалов А. А., Титаренко Л. А. Прикладная теория цифровых автоматов. — Донецк: ДонНТУ, Технопарк ДонНТУ УНИТЕХ, 2010. — 320 с.

4. Бибило П. Н. Основы языка VHDL: Учебное пособие. / П. Н. Бибило. — Изд. 5-е. — М.: Книжный дом «ЛИБРОКОМ», 2012. — 328 с.

5. Иваница С. В., Аноприенко А. Я. Особенности реализации операций тетралогики // Научные труды Донецкого национального технического университета. Серия: «Информатика, кибернетика и вычислительная техника» (ИКВТ-2011). Выпуск 13 (185). — Донецк: ДонНТУ, 2011. С. 134–140.

6. Anopriyenko A., Ivanitsa S., Hamzah A. Postbinary calculations as a machine-assisted realization of real interval calculations / International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE), 2 (4), July–August 2013, P. 91–94.

7. Аноприенко А. Я. Постбинарный компьютинг и интервальные вычисления в контексте кодо-логической эволюции. / А. Я. Аноприенко, С. В. Иваница — Донецк, ДонНТУ, УНИТЕХ, 2011. — 248 с.

8. Иваница С. В. Реализация арифметических операций сложения и вычитания над тетракодами / С. В. Иваница // Наукові праці Донецького національного технічного університету. Серія: «Проблеми моделювання та автоматизації проектування». № 1 (12)–2(13). — 2013. С. 149–158.

9. Аноприенко А. Я. Особенности реализации постбинарных логических операций / А. Я. Аноприенко, С. В. Иваница // Научно-теоретический журнал «Искусственный интеллект», № 2, 2011. С. 110–121.

10. 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) // Материалы II Всеукраинской научно-технической конференции студентов, аспирантов и молодых ученых — 11–13 апреля 2011 г. Т.2. Донецк, ДонНТУ. — 2011. С. 198–202.

11. Зубчук В. И., Сигорский В. П., Шкуро А. Н.. Справочник по цифровой схемотехнике. — К. Техника, 1990. — 448 с.

12. Баркалов А. А. Классические принципы организации и проектирования центрального процессора / А. А. Баркалов, Л. А. Титаренко, В. Н. Опанасенко. — Донецк: «Технопарк ДонНТУ УНИТЕХ», 2011. — 338 с.

13. Аноприенко А. Я. Иваница С. В., Бурлака Е. В. Программная реализация постбинарного кодирования интервалов // Научные труды Донецкого национального технического университета. Серия «Проблемы моделирования и автоматизации проектирования динамических

систем» (МАП-2011). Выпуск 10 (197): Донецк: ДонНТУ, — 2011. С. 207–214.

14. Бурлака Е. В., Иваница С. В., Аноприенко А. Я. Программная модель отображения тетракода на множествах действительных и интервальных чисел // Информатика и компьютерные технологии / Материалы VII международной научно-технической конференции студентов, аспирантов и молодых ученых — 22–23 ноября 2011 г., Т.1. Донецк, ДонНТУ. — 2011. С. 336–342.

15. Аноприенко А. Я., Иваница С. В., Кулибаба С. В. Представление постбинарных форматов чисел с плавающей запятой в контексте интервальных вычислений // Научные труды Донецкого национального технического университета. Серия: «Информатика, кибернетика и вычислительная техника» (ИКВТ-2011). Выпуск 14 (188). — Донецк: ДонНТУ, 2011. С. 44–49.

16. Яшкардин В. IEEE 754 — стандарт двоичной арифметики с плавающей точкой. http://softelectro.ru/ieee754.html.

17. Юровицкий В.М. IEEE754-тика угрожает человечеству — МФТИ, РГСУ, Москва. http://www.yur.ru/science/computer/IEEE754.htm.

18. Иваница С. В. Методы округления чисел с плавающей запятой, представленных в постбинарных форматах // Искусственный интеллект. ИИ-2012: материалы Международной научно-технической конференции (пос. Кацивели, АР Крым, 1–5 октября 2012 года). — Донецк: «Наука и образование», 2012. — С. 33–36.

19. Иваница С. В. Оценка погрешности представления вещественных чисел в постбинарных форматах с

плавающей запятой / С. В. Иваница // Научно-теоретический журнал «Искусственный интеллект», № 4. — Донецк, 2012. С. 32–44.

20. Steve Hollasch IEEE Standard 754 Floating Point Numbers, 2004. http://steve.hollasch.net/cgindex/coding/ieeefloat.html.

21. Аноприенко А.Я., Иваница С.В. Интервальные вычисления и перспективы их развития в контексте кодо-логической эволюции / А.Я. Аноприенко, С.В. Иваница // Научные труды Донецкого национального технического университета. Серия «Проблемы моделирования и автоматизации проектирования динамических систем» (МАП-2010). Выпуск 8 (168): Донецк: ДонНТУ, 2010. С. 150–160.

22. Назаров Н. Г. Метрология. Основные понятия и математические модели / Н. Г. Назаров — М.: Высшая школа, 2002. — 348 с.

23. Аноприенко А. Я. Пример Румпа в контексте традиционных, интервальных и постбинарных вычислений / В. А. Гранковский, А. Я. Аноприенко, С. В. Иваница // Научные труды Донецкого национального технического университета. Серия «Проблемы моделирования и автоматизации проектирования динамических систем» (МАП-2011). Вып. 9 (179): Донецк: ДонНТУ, 2011. С. 324–343.

24. Аноприенко А.Я., Иваница С.В. Гибкая разрядность и постбинарные форматы представления вещественных чисел // Вестник Инженерной Академии Украины. Теоретический и научно-практический журнал Инженерной Академии Украины. Выпуск 1. — Киев, 2012. С. 92–98.

25. Окулов С. М. Длинная арифметика. http://comp-science.narod.ru/DL-AR/okulov.htm.

26. System.Numerics Namespace. MSDN Library. .NET Framework 4. http://msdn.microsoft.com/en-us/library/system.numerics.aspx.

27. Леоненков А.В. Самоучитель по UML. [Электронный ресурс] — Режим доступа: http://khpi-iip.mipk.khar-kiv.edu/library/case/leon/index.html.

28. Ковалев А. А. Разработка постбинарного интервального АЛУ для модифицированных форматов чисел с плавающей запятой / А. А. Ковалев, С. В. Иваница, Л. И. Дорожко // «Інформаційні управляючі системи та комп'ютерний моніторинг» (ІУС КМ – 2014) // Збірка матеріалів V Всеукраїнської науково-технічної конференції студентів, аспірантів та молодих вчених — 22–23 квітня 2014 р., Донецьк, ДонНТУ. — 2014. С. 182–186.

29. Анопрієнко О. Я. Принцип роботи, структура і моделювання блоку перетворювача форматів у складі постбінарного співпроцесора / О. Я. Анопрієнко, С. В. Іваниця, С. В. Кулібаба // Міжнародний науково-технічний журнал «Інформаційні технології та комп'ютерна інженерія», № 1 (26). — Вінниця, 2013. — С. 59–65.

30. Лях, А.В., Иваница С. В., Аноприенко А. Я. Разработка программной библиотеки для реализации постбинарных вычислений // Инновационные перспективы Донбасса: материалы междунар. науч.-практ. конф., г. Донецк, 20–22 мая 2015 г. / ГВУЗ «ДонНТУ». — Донецк, 2015. — Т. 5: Компьютерные науки и технологии. — С. 32–37.

ПРИЛОЖЕНИЕ

Модифицированные форматы чисел с плавающей запятой

Модифицированные форматы чисел с плавающей запятой — форматы машинных чисел, аналогичные используемым в соврменных процессорах стандартным форматам (стандатт IEEE 754-2008). На рисунке ниже представлено соотношение модифицированных форматов (pbinary, или сокращенно pb) со стандарнтыми (binary) точности Ω с выделением полей знака, порядка, мантиссы, а также идентификатора формата.

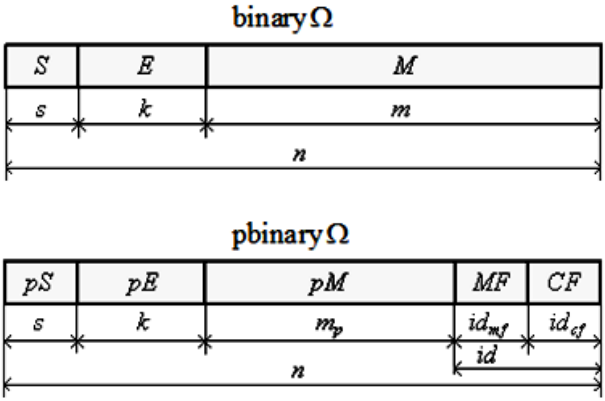


Рисунок — Структура числа в стандартном формате binaryΩ и в эквивалентном ему формате pbinaryΩ с указанием полей формата

Здесь *S* и *pS* — поля знака, *E* и *pE* — поля порядка со смещением, *M* и *pM* – поля остатка от мантиссы (без скрытых разрядов), *MF* — поле модификатора формата,

CF — поле кода формата. Размерность каждого поля выражена соответствующими строчными буквами, причем $n \equiv \Omega$ — количество разрядов поля 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 модифицированные 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 модифицированные 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 модифицированные 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 модифицированные 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.

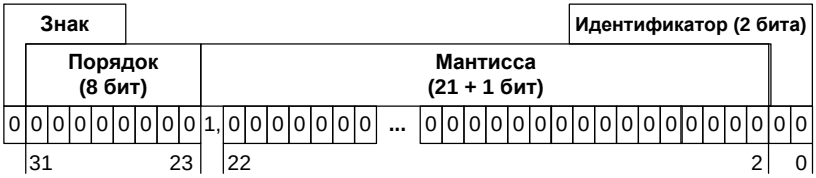
  По **точности представления** интервальные модифицированные постбинарные форматы делятся на:

  - PostBinary 64/16 ip;
  - PostBinary 128/32 ip;
  - PostBinary 256/64 ip.

PostBinary 32

Спецификация формата

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)	

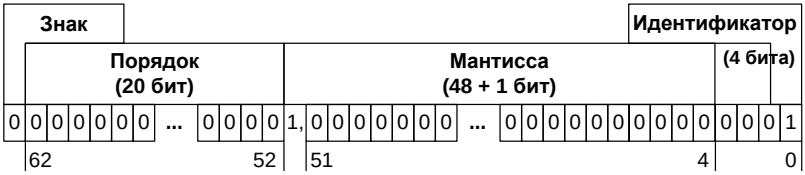
Числовые диапазоны:	**S Exp Mant MF+CF**	**Значение**
мининальное денормализованное	X 00000000 00000000000000000000001 00	≈ ±5.6051939e–45
максимальное денормализованное	X 00000000 11111111111111111111111 00	≈ ±1.1754938e–38
мининальное нормализованное	X 00000001 00000000000000000000000 00	≈ ±1.1754944e–38
максимальное нормализованное	X 11111110 11111111111111111111111 00	≈ ±3.4028229e+38
	(X – значение разряда 0 или 1)	

Относительная точность десятичных цифр: 6

PostBinary 64

Спецификация формата

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	XXXXXXXXXXXX...XX	0001
отрицательная бесконечность (–NaN)	1	11111111111	XXXXXXXXXXXX...XX	0001
	(где XXX..X – отличная от нуля мантисса)			

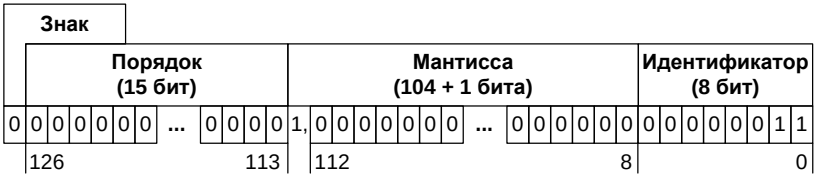
Числовые диапазоны:	**S**	**Exp**	**Mant**	**MF+CF**	**Значение**
минимальное денормализованное	X	00000000000	0000000000000...01	0001	≈ ±7.905050333499e–323
максимальное денормализованное	X	00000000000	1111111111111...11	0001	≈ ±2.2250738585072e–308
минимальное нормализованное	X	00000000001	0000000000000...00	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

PostBinary 128

Спецификация формата

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	00000000000000...00	00000011
отрицательный ноль	1	000000000000000	00000000000000...00	00000011
положительная бесконечность (+Infinity)	0	111111111111111	00000000000000...00	00000011
отрицательная бесконечность (–Infinity)	1	111111111111111	00000000000000...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)			

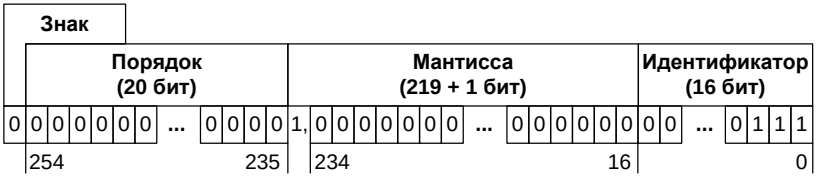
Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ± 1.65764483057613442839665633733063 e–4963
максимальное денормализованное	≈ ± 3.36210314311209350626267781732l6 e–4932
мининальное нормализованное	≈ ± 3.36210314311209350626267781732l8 e–4932
максимальное нормализованное	≈ ± 1.18973149535723176508575932662800 e+4932

Относительная точность десятичных цифр: 31–32

PostBinary 256

Спецификация формата

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**
минимальное денормализованное	≈ ± 1.828623360511354903950603959893667737399055001281 e–1578924
минимальное нормализованное	≈ ± 1.540612133552872047061497749015684650691656123143 e–157826
максимальное нормализованное	≈ ± 2.596370567831000776126596495726882827744734376348 e+157826

Относительная точность десятичных цифр:	66

PostBinary 32/16 f

Спецификация формата

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

PostBinary 64/32 f

Спецификация формата

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)

PostBinary 128/64 f

Спецификация формата

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)

PostBinary 256/128 f

Спецификация формата

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	≈ ± 1.65764483057613442839665633733063 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)

PostBinary 32/16 i

Спецификация формата

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**
минимальное денормализованное	≈ ± 1.192 e–7
максимальное денормализованное	≈ ± 6.092 e–5
минимальное нормализованное	≈ ± 6.104 e–5
максимальное нормализованное	±65 472
Относительная точность десятичных цифр:	3 (для левой и правой границ)

PostBinary 64/32 i

Спецификация формата

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**
минимальное денормализованное	≈ ±5.6051939 e–45
максимальное денормализованное	≈ ±1.1754938 e–38
минимальное нормализованное	≈ ±1.1754944 e–38
максимальное нормализованное	≈ ±3.4028229 e+38
Относительная точность десятичных цифр:	6 (для левой и правой границ)

PostBinary 128/64 i

Спецификация формата

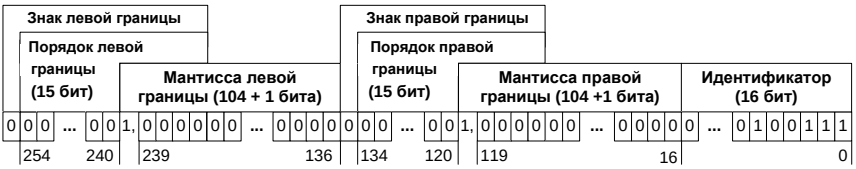
Формат **PostBinary 128/64 i** (**pb128/64i**) предназначен для хранения двух вещественных чисел (модификатор MF = 02h) **двойной точности** (код формата CF = 3h), представляющих собой **границы интервала**. Используется **бинарное (двоичное) кодирование**.

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	≈ ±2.2250738585072 9 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)

PostBinary 256/128 i

Спецификация формата

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**.

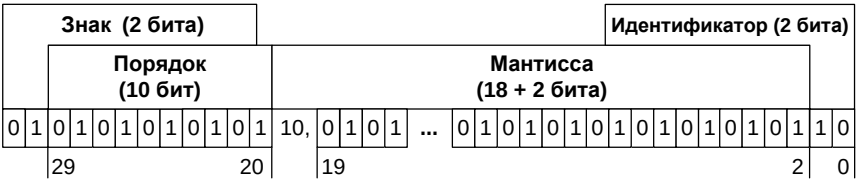


Размерность полей формата:	240 бит: 2 × (1 бит + 15 бит + 104 бит)
знак	1 бит
мантисса	15 бит
порядок	104 бит
Поле идентификатора:	16 бит (MF = 12 бит + CF = 4 бита)
значение модификатора формата MF	000000000010 – интервальный формат и бинарное кодирование
значение кода формата CF	0111 – обозначает поле данных длиной 256 бит
Смещение порядка:	+16383
Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ± 1.65764483057613442839665633733063 e–4963
максимальное денормализованное	≈ ± 3.36210314311209350626267781732l6 e–4932
мининальное нормализованное	≈ ± 3.36210314311209350626267781732l8 e–4932
максимальное нормализованное	≈ ± 1.18973149535723176508575932662800 e+4932
Относительная точность десятичных цифр:	31–32 (для леовй и правой границ)

PostBinary 32/16 p

Спецификация формата

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.

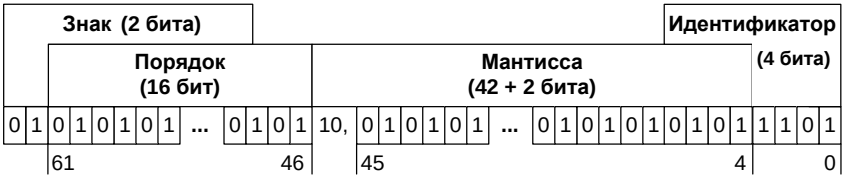


Размерность полей формата:	30 бит: 2 × (1 бит + 5 бит + 9 бит)
знак	2 бита
мантисса	10 бит
порядок	18 бит
Поле идентификатора:	2 бита (MF = 1 бит + CF = 1 бит)
значение модификатора формата MF	1 – числовой формат и постбинарное кодирование
значение кода формата CF	0 – обозначает поле данных длиной 32 бита
Смещение порядка:	+15
Числовые диапазоны:	**Значение**
минимальное денормализованное	≈ ± 1.192 e–7
максимальное денормализованное	≈ ± 6.092 e–5
минимальное нормализованное	≈ ± 6.104 e–5
максимальное нормализованное	±65 472
Относительная точность десятичных цифр:	3

PostBinary 64/32 p

Спецификация формата

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.

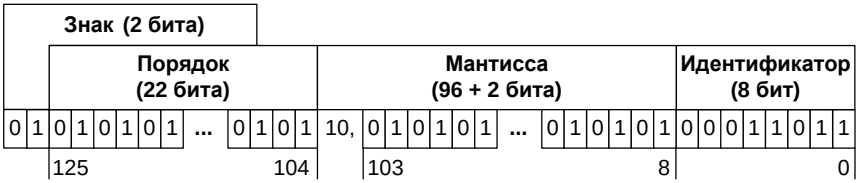


Размерность полей формата:	60 бит: 2 × (1 бит + 8 бит + 21 бит)
знак	2 бита
мантисса	16 бит
порядок	42 бита
Поле идентификатора:	4 бита (MF = 2 бита + CF = 2 бита)
значение модификатора формата MF	11 – числовой формат и постбинарное кодирование
значение кода формата CF	01 – обозначает поле данных длиной 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

PostBinary 128/64 p

Спецификация формата

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

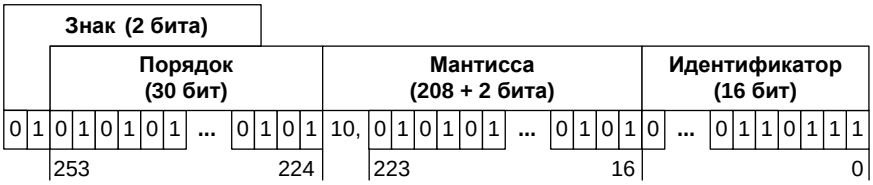
Числовые диапазоны:	**Значение**
мининальное денормализованное	≈ ±7.905050333499 e–323
максимальное денормализованное	≈ ±2.22507385850719 e–308
мининальное нормализованное	≈ ±2.225073858072 e–308
максимальное нормализованное	≈ ±1.7976931348623 e+308

Относительная точность десятичных цифр: 14–15

PostBinary 256/128 p

Спецификация формата

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	≈ ± 1.65764483057613442839665633733063 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

PostBinary 64/16 fp

Спецификация формата

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

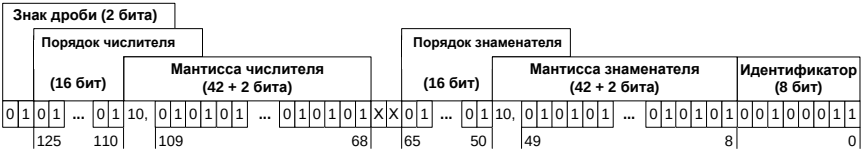
Числовые диапазоны:	**Значение**
минимальное денормализованное	≈ ± 1.192 e–7
максимальное денормализованное	≈ ± 6.092 e–5
минимальное нормализованное	≈ ± 6.104 e–5
максимальное нормализованное	±65 472

Относительная точность десятичных цифр: 3 (для числителя и знаменателя дроби)

PostBinary 128/32 fp

Спецификация формата

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)

PostBinary 256/64 fp

Спецификация формата

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)

PostBinary 64/16 ip

Спецификация формата

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)

PostBinary 128/32 ip

Спецификация формата

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**
минимальное денормализованное	≈ ±5.6051939 e–45
максимальное денормализованное	≈ ±1.1754938 e–38
минимальное нормализованное	≈ ±1.1754944 e–38
максимальное нормализованное	≈ ±3.4028229 e+38
Относительная точность десятичных цифр:	6 (для каждой границы интервала)

PostBinary 256/64 ip

Спецификация формата

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**
минимальное денормализованное	≈ ±7.905050333499 e–323
максимальное денормализованное	≈ ±2.22507385850719 e–308
минимальное нормализованное	≈ ±2.225073858072 e–308
максимальное нормализованное	≈ ±1.7976931348623 e+308
Относительная точность десятичных цифр:	14–15 (для каждой границы интервала)

Научное издание

Аноприенко Александр Яковлевич

Иваница Сергей Васильевич

ВВЕДЕНИЕ В ПОСТБИНАРНЫЙ КОМПЬЮТИНГ. АРИФМЕТИКО-ЛОГИЧЕСКИЕ ОСНОВЫ И ПРОГРАММНО-АППАРАТНАЯ РЕАЛИЗАЦИЯ

ISBN **978-966-8248-75-7**

Издательство: ООО «Технопарк ДонГТУ УНИТЕХ»

Свидетельство о внесении субъекта издательского дела в
государственный реестр издателей, изготовителей
и распространителей издательской продукции:
ДК № 1017 от 21.08.2002.
83000, г. Донецк, ул. Артема, 58, к.1.311
Тел.: (062) 304–90–19

Подписано к печати 24.06.2017. Формат 60x84 1/32
Усл. печ. л. 13,25. Печать лазерная. Заказ № 1706
Тираж 500 экз.

Отпечатано в типографии издательства
«Донецкая политехника»
Адрес: г. Донецк, ул. Артема, 58
Тел.: +380 (62) 304-60-82