Сергей Борисович Гашков
*Доктор физико-математических наук,
профессор кафедры дискретной математики
механико-математического факультета
МГУ им. М. В. Ломоносова.
Много лет преподавал математику также
в ФМШ N18 при МГУ (ныне СУНЦ МГУ).*

Математика внутри компьютера

Для того чтобы вывести
из ничтожества
всё, достаточно
единицы.
Готфрид Вильгельм Лейбниц

Все знают, что компьютеры помогают решать математические задачи. Многие думают, что математика без компьютеров практически уже не обходится, а некоторые даже полагают, что со временем компьютеры вообще заменят математиков (и не только их), когда появится искусственный разум. На самом деле в этих мнениях значение компьютеров для математики сильно преувеличено (хотя, возможно, есть в них и рациональное зерно). На самом деле многие математики используют компьютер только как печатную машинку (для того, чтобы писать тексты с формулами), изредка пользуются готовыми программами (Mathematica, Maple, Mathcad, Mathlab и др.) для вспомогательных вычислений и редко сами пишут программы для каких-то своих целей (если их не устраивают доступные программы, написанные другими).

Но мы не будем далее обсуждать затронутые темы, а рассмотрим совсем другую – как применяется математика в самом компьютере, т. е. не в написании программ для него, а непосредственно внутри него, а также при проектировании и производстве компьютеров (и другой электронной цифровой техники). С этой темой также связаны различные заблуждения. Часто полагают, что указанные вопросы относятся скорее к технике или, на крайний случай, к физике (и немного к химии), но не к математике. Мы постараемся на простых примерах показать, что внутри компьютеров (и вообще в любой цифровой технике) живёт и работает очень интересная математика, которая активно развивается вместе с развитием электронной техники.

Алгебра логики и компьютерная арифметика

Увы! То, что некогда возвышалось как монумент монотеизму, очутилось в чреве компьютера.
Т. Данциг

В наше время почти вся техника, связанная с передачей и обработкой информации, стала цифровой. Цифровыми стали аудио- и видеомагнитофоны, превратившись в MP3- и DVD-плейеры, айподы, телевизоры, фотоаппараты; а все виды электронно-вычислительной техники и мобильные телефоны были цифровыми изначально. Это означает, что информация, циркулирующая в этих устройствах, представляется (кодируется) в цифровом виде, т. е. как правило, в виде строк (последовательностей), состоящих из нулей и единиц. Этим строкам можно сопоставить по некоторым правилам целые числа, для чего обычно используется двоичная позиционная система их записи.

Таким образом, с определённой точки зрения все цифровые устройства генерируют потоки целых чисел[1], по некоторым правилам их преобразуют, обрабатывают, кодируют и декодируют и т. д. и т. п. и передают другим цифровым устройствам.

Существенную роль в этом играют алгоритмические процедуры, выполняющие арифметические и логические операции с различными типами числовых данных. Проектированием подобных алгоритмов и устройств, их реализующих, занимается компьютерная арифметика. Её математической основой является теория сложности функций алгебры логики.

В большей своей части компьютерная арифметика является двоичной арифметикой. Этому есть две причины. Во-первых, алгоритмы арифметических операций двоичной арифметики (арифметики, использующей двоичную позиционную систему) очень просты и являются в определённом смысле простейшими среди подобных алгоритмов для всех позиционных числовых систем. Во-вторых, дискретные электронные схемы как самые современные, так и использованные много лет назад, имеют в определённом смысле двоичную природу и легко описываются на языке алгебры логики. Алгебра логики применяется как для моделирования функционирования этих схем, так и для их проектирования (синтеза).

Булевы функции и логические функциональные элементы

Джордж Буль установил истинную связь алгебры и логики.
Август де Морган

Современные электронные схемы состоят из элементов, называемых также ячейками, которые имеют микроскопические размеры и представляют собой особые участки кремниевого кристалла. Физика и химия протекающих в них процессов весьма сложна и не будет здесь обсуждаться. Также сложна и технология их производства. Но логика работы этих элементов довольно проста, каждый из них реализует определённую логическую операцию.

Математической моделью простейших (однотактных) электронных схем являются логические функциональные элементы.

---

[1] Конечно, не только целых, но и «чисел с плавающей запятой», но сейчас для нас такие детали не существенны.

Функциональный элемент – это абстрактное устройство с одним или несколькими входами, на которые подаётся информация, и одним или несколькими выходами, информация с которых подаётся на входы других элементов или на выход всего большого устройства, содержащего этот элемент. Упомянутая информация содержится в состояниях входов или выходов элемента. Этих состояний у каждого полюса элемента (это совокупное название его входов и выходов) два, и они обозначаются нулём или единицей. Если состояния входов элемента определены (т. е. известны сопоставленные им значения 0 или 1), то значения на выходах элемента тоже однозначно определены, т. е. являются функциями значений входов.

Эти функции имеют аргументы, которые могут быть равными только 0 или 1, и сами эти функции принимают только значения 0 или 1. Такие функции называются *функциями алгебры логики*, или *булевыми функциями*. Последнее название дано в честь одного из первопроходцев математической логики, английского математика Джорджа Буля[1].

Говорят, что элемент реализует (на своих выходах) упомянутые булевы функции, однозначно определённые самим этим элементом.

Простейшим примером логического функционального элемента является элемент с одним входом и одним выходом, значение которого всегда противоположно значению входа. Это значит, что он реализует булеву функцию $f(x)$, определяемую равенствами $f(0)=1$, $f(1)=0$. Такую функцию называют *логическим отрицанием*, обычно обозначают символом $\neg x$ или чертой над символом переменной, а реализующий её элемент называют иногда *инвертором*.

Джордж Буль
(1815 – 1864)

Примерами элементов с двумя входами и одним выходом являются элементы, называемые иногда конъюнктором и дизъюнктором. Первый из них реализует функцию, называемую в алгебре логики *конъюнкцией*. Она принимает единичное значение только тогда, когда оба её аргумента равны единице (в остальных случаях она равна нулю). По существу она совпадает с обычной операцией умножения целых чисел, если её применять только к числам, равным нулю или единице. Поэтому её иногда называют логическим умножением и обозначают точкой, которую часто в записи формул опускают. Однако во многих случаях полезно подчеркнуть логическую природу этой функции и её тесную связь с логической связкой, также называемой конъюнкцией, и тогда для её обозначения используют символ &, который пишут между символами её аргументов, например

[1] Джордж Буль (1815 – 1864) – профессор университета в Корке (Ирландия). Сын сапожника, в математике был самоучкой. Основоположник математической логики. Отец шести дочерей, одна из которых стала математиком, другая – первой женщиной-профессором химии в Англии, а ещё одна – писательницей, известной под именем Этель Лилиан Войнич, автора романа «Овод». В СССР было издано двухтомное собрание её сочинений, в предисловии к которому упоминалось, что она любила русскую литературу, переводила её на английский и имела близких друзей среди русских революционеров-народовольцев.

$x\&y$, а не $\&(x, y)$, как следовало бы делать, придерживаясь функционального способа записи. В исчислении высказываний (разделе математической логики) символ конъюнкции, поставленный между двумя высказываниями, интерпретируется как союз «и», поэтому в англоязычных странах инженеры обозначают конъюнктор символом AND.

Другим примером элемента с двумя входами является дизъюнктор, реализующий булеву функцию, называемую *дизъюнкцией*. Эта функция является двойственной к конъюнкции и определяется двойственным образом, т. е. подобно конъюнкции, но с заменой 0 на 1 и наоборот. Более явно определение дизъюнкции можно дать следующим образом: дизъюнкция равна нулю тогда и только тогда, когда оба её аргумента нули. Для обозначения дизъюнкции используется символ $\vee$ (дизъюнкцию иногда называют логическим сложением, но нужно помнить, что в отличие от обычного сложения $1\vee 1$ равно 1, а не 2). В англоязычных странах дизъюнктор обозначают символом OR, так как функция дизъюнкция связана с логической связкой, также называемой дизъюнкцией, которая интерпретируется как союз «или». В русском языке, как и в английском, союз «или» обычно понимается в разделительном смысле, который отличен от того, как понимается этот союз в логике.

Двоичная система и логические операции

Не могу представить, чтобы кому-нибудь понадобилось
выполнять умножение со скоростью 4 000 операций в час.
Такое радикальное средство, как переход к восьмеричной системе,
не стоит навязывать человечеству ради нескольких личностей.
F.X. Wales, 1936

Произвольное натуральное число можно единственным способом представить в виде суммы степеней двойки, например $23 = 16 + 4 + 2 + 1$. Обозначая входящие в эту сумму степени единицами, а не входящие в неё степени нулями, можно кратко обозначить эту сумму булевым набором (в другой терминологии – вектором) $(10111)_2$. Индекс 2 напоминает о том, что число записано в двоичной системе. Единица, стоящая в младшем (самом правом) разряде, означает слагаемое 1, единица во втором справа разряде означает слагаемое 2, единица в третьем разряде означает 4, а нуль в четвёртом разряде означает отсутствие слагаемого 8, единица в пятом (старшем) разряде означает присутствие слагаемого 16 (в большинстве случаев разумно рассматривать только такие записи чисел в двоичной системе, в которых в старшем разряде стоит единица).

Главное достоинство двоичной системы – исключительная простота алгоритмов арифметических операций в ней. Таблица умножения в ней совсем не требует запоминания: любое число, умноженное на нуль, даёт нуль, а умноженное на единицу равно самому себе. Правило деления сводится к двум равенствам $0/1 = 0$, $1/1 = 1$, благодаря чему деление столбиком в двоичной системе делается проще, чем в десятичной, и по существу сводится к многократному вычитанию. Таблица сложения в двоичной системе чуть сложнее таблицы умножения (в отличие от десятичной системы), так как $1 + 1 = (10)_2$ и возникает перенос в следующий разряд.

В общем виде операцию сложения однобитовых чисел можно записать в следующем виде:

$$x + y = 2w + v,$$

где w, v – биты результата. Внимательно посмотрев на таблицу сложения, можно заметить, что бит переноса w – это просто произведение xy, потому что он равен единице, лишь когда x и y равны единице. Ясно также, что бит v равен $x+y$, за исключением случая $x=y=1$, когда он равен не 2, а 0. Операцию, с помощью которой по битам x, y вычисляют бит v, называют по-разному. Мы будем использовать для неё название *сложение по модулю* 2 и символ $\oplus$. Таким образом, сложение битов выполняется фактически не одной, а двумя операциями.

Логические тождества

Если остановить любого из этих прохожих, то вряд ли даже один из пятидесяти сможет внятно объяснить, что такое электричество, не говоря уже о каком-нибудь булевом логическом выражении.

Тибор Фишер,
«Идиотам просьба не беспокоиться», 2000

Указанные операции связаны между собой множеством тождеств, из которых мы приведём лишь следующие:

$$x\vee y = \neg(\neg x\neg y),\ xy = \neg(\neg x\vee\neg y),$$
$$x\oplus y = x\neg y \vee y\neg x,\ x \vee y = x\oplus y\oplus xy,$$
$$(x\vee y)z = xz\vee yz,\ (x\oplus y)z = xy\oplus xz,$$
$$xy\vee z = (x\vee z)(y\vee z),\ \neg x = x\oplus 1.$$

Задача. Проверьте эти тождества.

Иногда удобнее записывать отрицание длинной формулы в виде черты сверху, например, вместо $\neg(xyz)$ писать $\overline{xyz}$. Можно выразить указанные операции, называемые логическими, через обычные арифметические операции. Конъюнкция, например, просто совпадает с обычным умножением, поэтому её при записи часто обозначают точкой или вообще пропускают. Справедливы также равенства

$$x\oplus y = x + y - 2xy,\ x \vee y = x + y - xy.$$

Можно также выразить конъюнкцию и дизъюнкцию через операции взятия максимума или минимума:

$$x\vee y = \max(x,\ y) = 1-\min(1-x, 1-y),$$
$$xy = \min(x,\ y) = 1-\max(1-x, 1-y).$$

Логические операции и исчисление высказываний

Девушкам надо лишь уметь отличать единицу от нуля [...]

Нил Стивенсон,
«Система мира», 2011

Почему введённые выше операции называют логическими? Сопоставим каждому высказыванию A его «истинностное» значение $t(A)$: $t(A)=0$, если высказывание A ложно, и $t(A)=1$, если высказывание A истинно. Тогда истинностное значение составного высказывания из A и B можно вычислить через истинностные значения высказываний A, B по формулам: $t(A$ и $B) = t(A)\ \&\ t(B)$ и $t(A$ или $B) = t(A)\vee t(B)$.

В последней формуле мы предполагали, что высказывание «A или B» будет истинным и тогда, когда оба высказывания A и B истинны. Иногда союз «или» понимают в несколько другом, разделительном, смысле: составное высказывание «A или B» считается истинным только в случае, если ровно одно из высказываний A, B истинно, но не оба сразу. Но логическая связка, соответствующая разделительной дизъюнкции, в логике тоже есть, и соответствующая ей булева функция принимает единичное значение тогда и только тогда, когда один из её аргументов единица, а второй нуль. В алгебре логики её часто обозначают символом $\oplus$ и называют суммой по модулю два, так как она действительно совпадает с этой алгебраической операцией. В англоязычных странах реализующий её элемент обозначают символом XOR (eXclusive OR).

Эту функцию можно выразить через остальные три функции, например, следующим образом:

$$x \oplus y = (x \& \neg y) \vee (\neg x \& y).$$

В алгебре логики часто приходится выражать одни булевы функции через другие, причём во многих случаях это можно сделать множеством способов, которые порождают множество логических тождеств. Например, справедливо тождество

$$x \oplus y = (x \vee y)\ \neg(x \& y).$$

Краткая история алгебры логики и вычислительной техники

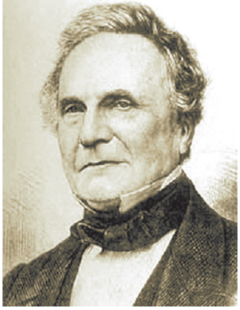
–А я смогу? Неужели это девушке под силу?
–Ты слыхала про леди Аду Байрон? – рассмеялся Мик. – Дочь премьер-министра и королева машин! Ада Байрон, ученица самого Беббиджа! Лорда Беббиджа, отца разностной машины и Ньютона современности!

Уильям Гибсон и Брюс Стерлинг, «Машина различий», 2002

Идею о возможности построить «логическую машину» впервые в достаточно чёткой форме высказывал знаменитый немецкий математик и философ Готфрид Вильгельм Лейбниц[1].

Годфрид Вильгельм Лейбниц (1646 – 1716)

Он же предлагал использовать в ней двоичную систему. Однако до реализации этой идеи у него руки не дошли по понятным причинам. Алгебры логики в то время, конечно, ещё не было, но Лейбниц выдвигал идеи и в этом направлении. Например, он писал, что когда-нибудь люди вместо того, чтобы спорить о том, правильно рассуждение или нет, просто сядут за вычисления, которые и решат спор.

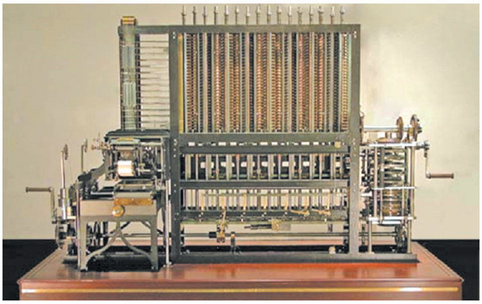


Чарльз Беббидж (1791 – 1871)

[1] Лейбниц является одним из героев серии фантастических романов Нила Стивенсона «Барочный цикл», недавно опубликованной в русском переводе. Герой этих романов Дэниел Уотерхауз, общий знакомый Лейбница и Ньютона (который тоже неоднократно появляется в этих романах), пытается построить логическую машину Лейбница (участвуя попутно в различных авантюрах).

Попытку реализовать идеи Лейбница предпринял в первой половине XIX в. английский математик и изобретатель Чарльз Беббидж. [1]

Он эти идеи ясно оформил и развил, добавив к ним много нового, и даже построил первый вариант этой машины.



Разностная машина Беббиджа

Однако в то время физика электрических явлений была в недостаточно развитом состоянии (хотя современник и соотечественник Беббиджа Майкл Фарадей уже сделал свои замечательные открытия в области электромагнетизма), и Беббидж решил строить свою машину из механических деталей. Понятно, что её быстродействие не могло быть высоким. Но проблема была ещё в том, что эти детали надо было изготовлять вручную с высокой точностью, работе машины мешало трение и прочие неприятные механические эффекты. Преодолеть эти трудности Беббидж не смог, что совершенно не удивительно. Его проект явно опережал своё время на сотню лет и был реализован (на другой технической базе) в 40-е гг. XX в. в США, где был построен первый электромеханический компьютер, а потом и компьютер на электронных лампах. Любопытно, что два человека, имена которых скорее всего вспоминают в связи с историей развития компьютеров, – Буль и Беббидж – жили в одной стране и в одно время, знали о существовании друг друга, но совершенно не предполагали, что сферы их деятельности будут пересекаться в будущем и их достижения будут упоминаться рядом, тем более что Беббидж, хотя и предполагал использовать в своей машине двоичную систему, совершенно не подозревал о возможности использования алгебры логики для описания её работы, так как в гораздо большей степени для этого была нужна точная механика.

После Буля к исследованиям в области алгебры логики присоединились и другие учёные (увы, Беббидж не заинтересовался логикой). Среди них можно назвать Августа де Моргана и Чарльза Додж-

[1] Интересно, что Беббидж, а также его светская знакомая Ада Августа Лавлейс, написавшая статью, популяризировавшую работы Беббиджа, в которой она объясняла, как можно использовать его машину, благодаря чему и вошла в историю как «первая программистка», и её отец, знаменитый поэт лорд Байрон, стали героями романа «Машина различий», написанного У. Гибсоном и Б. Стерлингом в модном жанре альтернативной фантастики (и одновременно киберпанка). В этом романе описан мир, в котором научно-техническая революция началась в Англии в 20-е гг. XIX века. Лорд Байрон, вместо того, чтобы поддерживать разрушителей машин – луддитов – и потом уехать воевать за освобождение Греции, стал премьер-министром, а мозговым центром этой революции стал, естественно, лорд Беббидж (в реальности он лордом не был). Не отставали и французы – в романе упоминается “мощнейший вычислитель французской академии «Император Наполеон»”. Впрочем, основная интрига закручивается вокруг вымышленных персонажей, и машины Беббиджа играют в ней незначительную роль, хотя и дали название роману, по-английски звучащему как «The Difference Engine» – разностная машина (так и назывался в реальности первый вариант машины Беббиджа). Трудно сказать, то ли переводчик не знал об этом и неправильно перевёл название романа, то ли выбрал такой перевод умышленно, как содержащий тонкую игру смыслов.

сона, более известного под псевдонимом Льюис Кэрролл.[1]

Россия также имеет давние традиции исследований в области алгебры логики. Существенный вклад в её развитие внёс казанский астроном Платон Сергеевич Порецкий (1846 – 1907).

В его работах, например, фактически впервые появились дизъюнктивные и конъюнктивные нормальные формы булевых функций[2]. Он их использовал для решения булевых уравнений.

Порецкий Платон Сергеевич (1846 – 1907)

Идея о применении алгебры логики для моделирования работы электрических или электромеханических схем впервые была высказана в начале прошлого века известным голландским физиком Паулем Эренфестом[3]. Развивать эту идею дальше он не стал, и её забыли до конца тридцатых годов, когда она была одновременно и независимо друг от друга опубликована тремя учёными: в СССР – научным сотрудником физфака МГУ В.И. Шестаковым[4], в США – инженером и математиком К. Шенноном и в Японии – инженером-электриком А. Накасимой. По-видимому, первым был Шестаков, но его диссертация была опубликована спустя несколько лет после её написания и защиты, а Шеннон опубликовал свою статью раньше.

Шестаков Виктор Иванович (1907 – 1987)

Шеннон рассматривал задачу моделирования работы электромеханических устройств, построенных из реле, и предложил в качестве модели понятие контактной схемы.

Клод Шеннон (1916 – 2001)

[1] Кстати, Кэрролл кроме всемирно известных сказок написал и несколько замечательных научно-популярных книг, например «Историю с узелками», которую неоднократно издавали в русском переводе. В этой книге есть большой раздел, посвящённый логическим задачам и головоломкам, в котором Кэрролл развил свой собственный аппарат, ориентированный на решение подобных задач.

[2] Американец Блэйк, занимавшийся дизъюнктивными нормальными формами в 30-е гг. XX в., в своей диссертации ссылался на работы Порецкого.

[3] Кстати, долгое время жившим в России. Его женой была математик Татьяна Афанасьева. У них была дочь, тоже Татьяна и тоже математик.

[4] Виктор Иванович Шестаков (1907 – 1987) – советский логик и теоретик-электротехник.

Если реле пропускало ток, то соответствующий ему контакт являлся замкнутым и его состояние описывалось символом 1, а если нет, то контакт становился разомкнутым и его состояние описывалось символом 0. С тех времён элементная база электронно-вычислительной техники полностью сменилась несколько раз, пройдя путь от электронных ламп довольно больших размеров до миниатюрных транзисторов, представляющих собой зоны кремниевого кристалла, размеры которых измеряются нанометрами и которых на кристалле может располагаться несколько миллионов. Такие электронные схемы в начале их появления назывались *интегральными схемами*, потом большими интегральными схемами (английская аббревиатура LSI) и, наконец, сверхбольшими интегральными схемами (аббревиатура VLSI).

Физика и химия происходящих в электронных схемах явлений также менялась вместе с изменением принципов их работы, но по-прежнему для их математического моделирования используется аппарат алгебры логики, только вместо контактных схем (которые в определённых ситуациях также используются) применяются логические схемы из функциональных элементов. Разумеется, для изучения тонких аспектов поведения реальных электронных схем (например, для вычисления их различных числовых характеристик, скажем, силы протекающих в них токов, потребляемой мощности, температуры различных зон) приходится применять сложные физические модели, использующие, в частности, дифференциальные уравнения, но для моделирования процессов обработки информации, происходящих в схеме, применяется алгебра логики. Конечно, с определённой точки зрения алгебро-логическая модель функционирования электронной схемы является грубым приближением (низкий потенциал обозначают нулём, высокий – единицей), но это приближение адекватно описывает работу схемы, и без его применения ни анализ, ни синтез сколь-либо сложных схем был бы невозможен.

Логические схемы компьютерной арифметики

> В главе, посвящённой Пуницланскому институту, я описал логическую схему И-НЕ, которая используется в пуницланских компьютерах, но не рассказал о том, как она может применяться в качестве составной части более сложных схем.
>
> *Александр Дьюдни, «Планиверсум», 2010*

Если у нас нет в распоряжении элемента XOR, но есть элементы OR, AND и NAND (так часто обозначают элемент, реализующий отрицание конъюнкции, используется также аналогичное обозначение NOR), то можно построить логическую схему из функциональных элементов, реализующую функцию XOR. Эта схема состоит из трёх элементов, соединённых друг с другом, как показано на рис. 1.

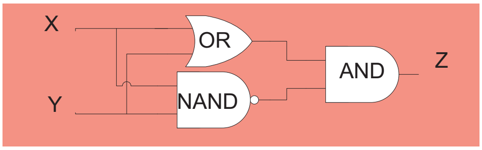


Рис. 1. Схема, реализующая функцию «Исключающее или»

Символы X и Y в этой схеме обозначают её входы, теми же символами удобно обозначать булевы переменные (т. е. переменные, принимающие значения 0 или 1), значения которых определяют состояния этих входов в рассматриваемый момент. Символ Z обозначает её выход. Если состояния входов схемы описываются значениями X и Y (иногда говорят, на входы поданы значения X и Y), то на выходе элемента OR появляется значение $X \vee Y$, на выходе элемента NAND – значение $\neg(XY)$, а на выходе элемента AND (он же является выходом всей схемы) – значение $(X \vee Y)(\neg XY)$. Согласно указанному выше тождеству это значение совпадает с $X \oplus Y$. Поэтому данная схема реализует функцию

$$f(x, y) = x \oplus y.$$

Неформально говоря, схема из функциональных элементов есть произвольный способ соединения выходов некоторых элементов с входами других элементов, в котором не появляются замкнутые циклы. Наличие таких циклов в реальной микросхеме приведёт к её неустойчивой работе в некоторых ситуациях, а наличие циклов в логической схеме сделает невозможным корректное определение её функционирования и, в частности, определение булевых функций, реализуемых выходами схемы.

Схемы из функциональных элементов имеют такое длинное название, чтобы их отличать от других видов схем, также предназначенных для реализации булевых функций, например, контактных схем. Так как другие виды схем далее не рассматриваются, то для краткости будем называть схемы из функциональных элементов просто схемами (в теоретическом программировании известно равносильное понятие, называемое неветвящейся программой).

Что такое сложность схемы

Теперь он понял, что просто устроенная машина
способна производить бесконечно сложный результат.
Нил Стивенсон,
«Криптономикон», 1999

Размером (или *сложностью*) схемы называется число составляющих её элементов. В примере, рассмотренном на рис. 1, сложность равна 3.

В практических приложениях реальные схемы размещены на кремниевом кристалле, и под размером схемы естественно понимать её площадь. Площадь схемы определяется суммарной площадью составляющих её элементов и площадью, занимаемой проводами, соединяющими элементы. Задача минимизации площади схемы является важной прикладной задачей (в последнее время её актуальность снизилась, так как современные технологии позволяют укладывать на кристалл схемы из огромного числа элементов).

Оценить долю площади, занимаемой проводами, довольно сложно. Она сильно зависит от используемого алгоритма укладки элементов схемы, называемого в англоязычной литературе *placement*, и от алгоритма разводки, или трассировки проводов, по-английски называемого *routing*. Хотя у сложных схем она может быть довольно велика, грубо её можно оценить как сумму площадей составляющих схему элементов. Поэтому иногда используется следующее обобщение понятия сложности схемы: *сложность схемы* – это сумма размеров (или весов) составляющих её элементов (предполагается, что каждому элементу сопоставлено число, называемое весом или размером элемента). Как правило, чем больше сложность, тем больше площадь.

В примере на рис. 1 выход каждого элемента присоединяется ко входу только одного другого элемента (говорят, что элементы схемы не имеют ветвлений), и схема имеет только один выход, тем самым реализуя только одну функцию. Такие схемы часто называют *формулами*. Входы схемы в рассмотренном примере имеют ветвления (каждый из них присоединяется ко входам двух элементов). Формулы, у которых и входы не имеют ветвлений, называют *бесповторными формулами*. Возможности таких формул для реализации булевых функций весьма ограничены.

Рассмотренная схема построена из элементов OR, AND, NAND. Кружок при элементе NAND изображает инвертор (обозначаемый в англоязычной литературе NOT), но на самом деле технология такова, что элемент NAND составляется из отдельных частей, называемых в англоязычной литературе *gate*, даже более просто, чем элемент AND, и имеет меньшую площадь и задержку при сравнимых электрических характеристиках.

Набор различных элементов, использованных при построении схемы, называется её *базисом*. В теоретических исследованиях обычно используются базисы {OR, AND, NOT}, {OR, XOR, AND, NOT} или базис из всех двухвходовых элементов. На практике базис, который можно использовать при построении схем, обычно определён заранее и известен разработчику. В инженерной терминологии он называется *технологической библиотекой*.

Современные библиотеки содержат сотни элементов (в англоязычной литературе вместо термина «элемент» используют термин *cell*, что переводится как «ячейка»), элементы могут иметь три, четыре, а иногда и более входов, иногда два или несколько выходов, логически одинаковые элементы могут иметь разные модификации, имеющие разные значения площади, задержки и разные электрические характеристики.

Полусумматор – схема для сложения двух битов

Будь силён в расчётах, и часу не проводи
без повторения их, ибо математика – наука свирепая.
«Кабус-намэ», XI в.

Для выполнения сложения двух однобитовых чисел обычно делают специальный логический элемент с двумя входами X, Y и двумя выходами U, V, как бы составленный из элемента конъюнкции и элемента сложения по модулю два. Этот элемент часто называют *полусумматором* (по-английски *half adder*) и обозначают HA (или FA2). Реализующая его схема представлена на рис. 2.

Задача. Постройте в базисе {AND, OR, NOT} схему для полусумматора сложности 4.

Указание. Воспользуйтесь формулой $x \oplus y = (x \vee y) \& \neg(x \& y)$.

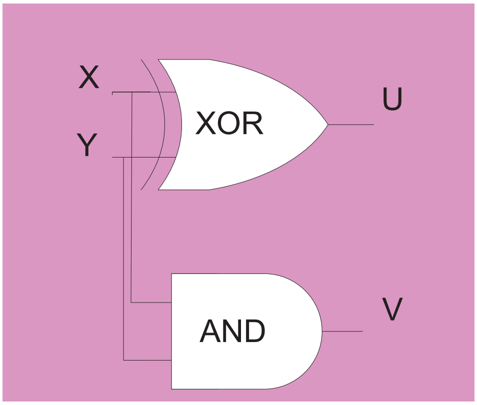


Рис. 2. Полусумматор (схема для сложения двух битов)

Глубина схемы и её задержка

Всё следует делать настолько простым,
насколько это возможно, но не проще.
Альберт Эйнштейн

Другой важной характеристикой схемы является её глубина. *Глубиной схемы* называется максимальное число её элементов, образующих цепь, соединяющую какой-либо вход схемы с одним из её выходов. Например, у схемы на рис. 2 глубина равна единице, а у схемы на рис. 1 – двум.

Задержкой схемы называется время, прошедшее с момента появления сигнала на входах схемы до момента появления сигнала на её выходе. (Как правило, значения входов схемы стабилизируются в разные моменты времени, и тогда отсчёт начинается с того, чьё значение стабилизировалось последним.)

Глубина – не менее важная характеристика схемы, чем её сложность. Сложность логической схемы в значительной степени определяет площадь соответствующей реальной схемы, расположенной на кремниевом кристалле. Глубина же логической схемы в значительной мере определяет задержку реальной схемы. Как правило, чем больше глубина, тем больше задержка.

Сложность схемы часто не имеет существенного значения, так как современные технологии позволяют разместить на кристалле очень большие схемы. А минимизация задержки схемы очень важна, так как задержка комбинационной части многотактной схемы определяет её тактовую частоту – чем меньше задержка, тем выше частота.

Теоретически вычислить задержку реальной схемы очень сложно. Цепей элементов схемы, соединяющих её входы с выходами (эти цепи также называют путями), обычно довольно много, и задержка схемы определяется задержкой по самому плохому в определённом смысле пути, который называется критическим. Задержка по данному пути определяется не только суммой задержек всех элементов, лежащих на этом пути. Следует учитывать также задержку соединяющих эти элементы проводов. Задержка элемента зависит от того, между каким его входом и каким его выходом она измеряется, а также от электрических характеристик самого элемента и элементов, непосредственно с ним связанных в рассматриваемой схеме, она зависит от температуры схемы и даже от того, какие логические значения подаются в рассматриваемый момент на входы этого элемента и изменяется ли (и в какую сторону) значение на его выходе.

Тем не менее, хотя и не очень точно, задержку пути можно оценить как сумму задержек его элементов. Если задержки всех элементов равны, то эта величина, как правило, определяется глубиной схемы[1]. Разумеется, понятие глубины схемы можно расширить, допустив, что элементы базиса могут иметь произвольные неотрицательные задержки, а глубину цепи элементов определить как сумму их задержек.

---

[1] Имеется замеченное В. М. Храпченко тонкое различие между глубиной и задержкой даже в предположении, что задержки всех элементов равны единице, а задержкой проводов и прочими эффектами пренебрегаем. Но здесь мы не можем касаться этого вопроса.

Схема для сложения трёх битов

В мире есть много трудных вещей, но нет
ничего труднее четырёх действий Арифметики.
Беда Достопочтенный (673 – 735)

На рис. 3 приводится более сложный пример схемы, в которой есть и ветвление элементов, и два выхода, которые не присоединяются ни к каким входам других элементов (но выход элемента, который присоединяется к входу другого элемента, при желании тоже может быть объявлен выходом схемы).

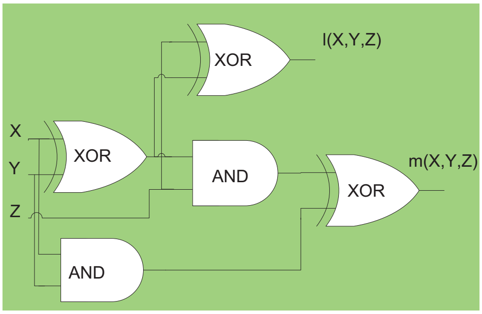


Рис. 3. Схема для сложения трёх битов

В этой схеме входы обозначены соответствующими им символами булевых переменных X, Y, Z, а на двух её выходах реализуются функции

$$l(x,\ y,\ z) = x\oplus y\oplus z\ –$$

сумма трёх переменных по модулю два (называемая также линейной функцией трёх переменных или счётчиком нечётности с тремя переменными) – и функция

$$m(x,\ y,\ z) = xy\vee xz\vee yz,$$

называемая функцией голосования комитета из трёх человек, мажоритарной функцией, или короче – *медианой*.

Первая из этих функций может быть определена следующим словесным описанием. Её значение по аргументам x, y, z вычисляется как остаток от деления обычной суммы $x + y + z$ на 2. Равносильным образом можно определить эту функцию формулой

$$(x\oplus y)\oplus z.$$

Медиану можно определить следующим словесным описанием. Она равна единице тогда и только тогда, когда не менее двух из её аргументов единицы. Равносильным образом её можно определить формулой

$$(xy\vee xz)\vee yz$$

или чуть более короткой формулой

$$x(y\vee z)\vee yz.$$

Не так очевидно, что ту же функцию можно определить формулами

$$xy\oplus(xz\oplus yz) = x(y\oplus z)\oplus yz.$$

Рассмотренная схема (её в англоязычной литературе называют *Full Adder* и обозначают FA3) называется *схемой сложения трёх битов* потому, что она действительно выполняет сложение трёх одноразрядных чисел в двоичной системе по следующим формулам:

$$x + y + z = 2u + v,\ u = m(x,y,z) =$$
$$= (x\oplus y)z\oplus xy,\ v = l(x,y,z) = (x\oplus y)\oplus z.$$

Для непосредственной проверки этих формул нужно перебрать все 8 значений булевого трёхмерного вектора $(x,\ y,\ z)$ от (0,0,0) до (1,1,1). Но если учесть, что функции $l(x, y, z)$ и $m(x,\ y,\ z)$ симметрические, т. е. не зависят от порядка переменных, то перебор сокращается до четырёх следующих вариантов: среди чисел x, y, z нет единиц, есть ровно одна единица, ровно две единицы, ровно три единицы. При этом следует заметить, что результат сложения трёх битов в двоичной системе в рассматриваемых случаях находится следующим образом:

$$0+0+0 = 0 = (00)_2,\ 1+0+0 = 1 = (01)_2,$$
$$1+1+0 = 2 = (10)_2,\ 1+1+1 = 3 = (11)_2.$$

Задача. Построить в базисе {AND, OR, NOT} схему сумматора трёх битов сложности 10.

Указание. Воспользуйтесь формулами

$$x\oplus y=(x\vee y)\neg(xy),$$
$$l(x,y,z) = (x\oplus y)\oplus z =$$
$$= ((x\oplus y)\vee z)\neg((x\oplus y)z),$$
$$m(x,y,z)= (x\vee y)z\vee xy.$$

Двоичные сумматоры – схемы для сложения двоичных чисел

Регистры редко применялись поодиночке.
Обычно они громоздились один на другой [...]
Нил Стивенсон,
«Криптономикон», 1999

Сложение двух n-разрядных чисел $(x_n, ..., x_1)_2$ и $(y_n, ..., y_1)_2$, как и в десятичной системе, приводит к появлению переносов в следующий разряд, которые необходимо учитывать в вычислении. Эти переносы также равны нулю или единице (если перенос равен нулю, то в ручном вычислении он фактически не выполняется, но логическая схема обязана правильно работать и в этом случае, ведь она не знает, какой перенос пришёл из предыдущего разряда). Обозначим перенос из $(i–1)$-го разряда в следующий i-й разряд через w_i ($w_1 = 0$, потому что предыдущего разряда в этом случае просто нет). Тогда для вычисления z_i (i-го бита результата) нужно сложить биты x_i и y_i и бит переноса w_i. Это сложение выполняем по формулам

$$x_i + y_i + w_i = 2v_i + u_i,\ v_i = m(x_i, y_i, w_i),$$
$$u_i= l(x_i, y_i, w_i)$$

с помощью схемы FA3. Тогда

$$z_i = u_i = l(x_i, y_i, w_i),$$

а следующий бит переноса

$$w_{i+1} = v_i = m(x_i, y_i, w_i).$$

При сложении n-разрядных чисел получается, вообще говоря, $(n+1)$-разрядное число. Его старший бит $z_{n+1} = w_{n+1}$ равен последнему переносу.

Задача. Докажите, что переносы действительно вычисляются по указанным формулам.

Схема сложения трёхразрядных чисел приведена на рис. 4. Аналогичным образом выглядит и схема сложения n-разрядных чисел.

Задача. Докажите, что сложность указанного n-разрядного сумматора в базисе {AND, OR, NOT, XOR}

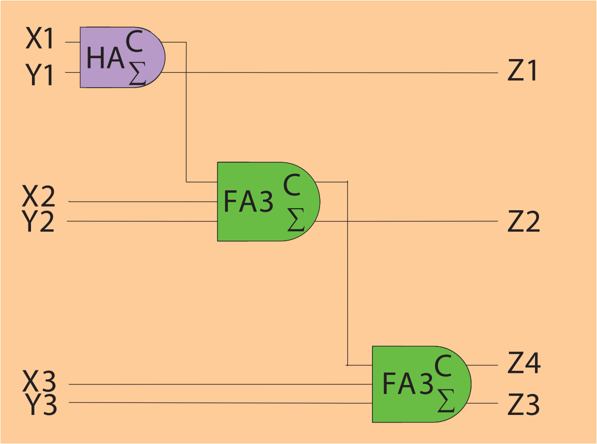


Рис. 4. Последовательный сумматор трёхразрядных чисел

равна $5n–3$, а в базисе {AND, OR, NOT} – $10n–6$.
Н.П. Редькин[1] доказал, что сумматоров меньшей сложности в базисе {AND, OR, XOR, NOT} не существует. Построенный сумматор является поэтому *минимальной схемой*. Но у этой схемы есть существенный недостаток – она имеет большую глубину. Её глубина в базисе {AND, OR, XOR} равна $2n–1$ при подходящем выборе схем, реализующих модули FA3.

Задача. Проверьте, что это действительно так.

Литература

1. *Гашков С.Б.* Системы счисления и их применение. – М.: МЦНМО, 2012.
2. *Гашков С.Б.* Занимательная компьютерная арифметика. Математика и искусство счёта на компьютерах и без них. Либроком, 2012.
3. *Гашков С.Б.* Занимательная компьютерная арифметика. Быстрые алгоритмы операций с числами и многочленами. Либроком, 2012.
4. *Петцольд Ч.* Код. – М.: Издательско-торговый дом «Русская редакция», 2001.
5. http:// lomonosov-fund.ru/enc/ru/encyclopedia:0135751
6. http:// lomonosov-fund.ru/enc/ru/encyclopedia:0135750
7. http:// lomonosov-fund.ru/enc/ru/encyclopedia:0135754

[1] Николай Петрович Редькин – профессор кафедры дискретной математики мехмата МГУ, известный специалист в области синтеза и тестирования логических схем.

Мудрые мысли Мудрые мысли Мудрые мысли

Человек – это дробь, у которой в числителе – его действительные возможности, а в знаменателе – его мнение о себе.

Л.Н. Толстой

Старайся стать разумным, а не богатым. Богатства можно лишиться, разумность всегда с тобой.

Эзоп

Образованность можно скрыть ото всех, невежество – только от себя.

В.З. Черняк

Основная ошибка, которой следует остерегаться, – полагать, что мы знаем больше, чем на самом деле.

Сократ

Таланты истинны на критику не злятся,
Их повредить она не может красоты –
Одни поддельные цветы
Дождя боятся.

И.А. Крылов

В будущем неграмотным будет считаться не тот, кто не умеет читать, а тот, кто не умеет обучаться.

Э. Тофлер