**Филиппов Константин Сергеевич**
*Инженер и преподаватель ГОУ «Многопрофильный технический лицей №1501».*

# Логика «железная», а с арифметикой что?

Примитивные приспособления для счёта известны ещё с глубокой древности. А первые механические счётные машины появились только в XVII веке и прослужили человечеству верой и правдой вплоть до середины XX века. Вычисления проводились в привычной всем десятичной системе. Открытие электричества усовершенствовало конструкцию этих машин лишь тем, что механизмы стали приводиться в движение уже не мускульной силой, а электроприводами. Революционными в подходе к созданию вычислительных машин стали принципы, сформулированные венгро-американским математиком Джоном фон Нейманом в 1946 году. Один из них – двоичное кодирование информации.

Вычислительная машина проводит операции не в десятичной, а в двоичной системе, что, безусловно, технически проще. Представьте себе на минутку, что в компьютере реализованы вычисления в десятичной системе. Как в этом случае можно интерпретировать десятичные цифры? Поскольку машина электрическая, логично сопоставить цифрам уровни напряжения. Например: «1» – один вольт; «2» – два вольта и т. д. И тут возникает серьёзная проблема: для определения цифры требуется измерить напряжение, причём достаточно точно. Следовательно, к каждому контакту арифметического устройства должен быть подключён вольтметр. И наоборот: как преобразовать цифру в напряжение соответствующего уровня? Задача технически решаемая. Вопрос – какой ценой?! Дальше – больше: реализовать хотя бы элементарные арифметические действия… Страшно представить, каких размеров получится

этот вычислительный аппарат и сколько он будет стоить!

Куда проще реализовать вместо десяти только два состояния электрической цепи: «включено» (есть напряжение) и «выключено» (нет напряжения). Определить лишь наличие электрического напряжения технически намного проще, чем измерять его величину.

Электронные схемы, работающие с такого рода сигналами, конструктивно просты и компактны. В частности, в качестве основного элемента статической памяти компьютера выступает *синхронный D-триггер*. Триггеры продолжительное время могут пребывать в одном из двух устойчивых состояний (включён или выключен), тем самым запоминая двоичную информацию. На рис. 1 представлено условное обозначение $D$-триггера (его ещё называют «защёлка»).

Его внутренняя схема может быть реализована по-разному, но, как правило, основные составляющие схемы – транзисторы. (Современные технологии производства интегральных микросхем (ИС, ИМС) позволяют разместить миллионы транзисторов всего лишь на одном

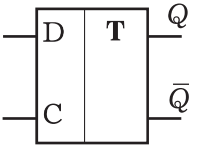



Рис. 1

квадратном миллиметре площади кристалла полупроводника!) Принцип действия прост: при поступлении электрического импульса на вход синхронизации $C$ значение сигнала на входе $D$ «запоминается» в триггере: на верхнем (прямом) выходе $Q$ устанавливается тот же сигнал, что был на входе $D$, а на нижнем (инверсном) выходе $\bar{Q}$ – ему противоположный. Таким образом, один триггер хранит один бит информации: «1» или «0».

## «Железная» логика

Возможно, в научно-популярных и даже художественных изданиях и фильмах, где упоминается вычислительная техника, вам встречались такие названия устройств как счётчики, дешифраторы, регистры и т. п. Без них не обходится ни одна современная вычислительная система. Но, если приглядеться, вся эта сложная компьютерная логика основывается лишь на трёх элементах: «НЕ», «И», «ИЛИ». Они реализуют три логических операции: отрицание (инверсия), умножение (конъюнкция), сложение (дизъюнкция). То есть компьютер на самом деле «умеет» выполнять не арифметические, а логические операции. Поэтому выше и было упомянуто словосочетание «компьютерная логика» (никак не арифметика). И всё же, компьютер должен считать. А стало быть, «логику» надо как-то привести к «арифметике». Но для начала вспомним правила выполнения основных логических операций.

Далее рядом с таблицей истинности операции приведено изображение логического элемента, который выполняет указанную операцию (рис. 2–4).

| $A$ | $\bar{A}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

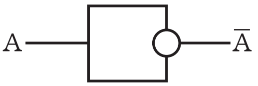



Рис. 2. Инверсия

| $A$ | $B$ | $A \wedge B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

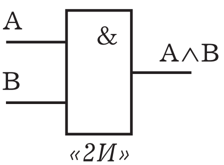



Рис. 3. Конъюнкция

| A | B | A∧B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

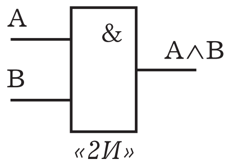



Рис. 4. Дизъюнкция

Обозначения: «0» – логический ноль, «ложь», низкий (около нуля вольт) уровень электрического напряжения; «1» – логическая единица, «истина», высокий уровень (чуть менее величины напряжения питания элементов схемы). Цифра «2» в названии элементов обозначает количество входов. Их может быть и больше.

Часто при разработке схем используют элементы с инверсным выходом (рис. 5, 6).

| A | B | *не* (A∧B) |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

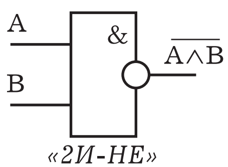



Рис. 5

| A | B | *не*(A∨B) |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

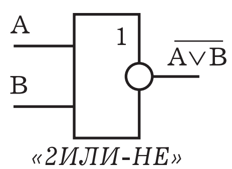



Рис. 6

По законам общей инверсии (законы де Моргана):

$$\overline{A \wedge B} = \overline{A} \vee \overline{B}, \quad \overline{A \vee B} = \overline{A} \wedge \overline{B}.$$

Если вспомнить упомянутый выше $D$-триггер, то его *логическая* реализация может быть, например, такой, как на рис. 7.

При включении электропитания схемы и отсутствии сигнала логической единицы на входе синхронизации $C$ в точках $S$ и $R$ установятся значения логической единицы независимо от того, какой сигнал подаётся на вход $D$. При этом на выходах

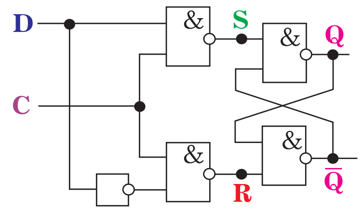



Рис. 7

$Q$ и $\overline{Q}$ установится произвольное (случайное) устойчивое состояние «$Q = 0$, $\overline{Q} = 1$» или «$Q = 1$, $\overline{Q} = 0$». (Конкретное состояние определяется быстродействием электронных схем, на которых построены логические элементы.) Пусть для определённости на выходе $Q$ установился «0», а на выходе $\overline{Q}$ – «1». В этом случае на вход *правого верхнего* элемента «2И-НЕ» с выхода нижнего элемента поступает сигнал «1», который в сочетании с единицей на входе $S$ будет поддерживать «0» на выходе $Q$. В свою очередь, «0» с выхода $Q$ поступает на вход *правого нижнего* элемента «2И-НЕ», и в сочетании с единицей на входе $R$ будет поддерживать «1» на выходе $\overline{Q}$. В результате триггер будет пребывать в таком устойчивом состоянии сколь угодно долго. Изменение сигнала на входе $D$ повлияет на состояние триггера только в случае прихода на вход $C$ короткого электрического импульса «1» – *сигнала синхронизации*.

Рассмотрим процесс переключения триггера из состояния «$Q = 0$, $\overline{Q} = 1$» в противоположное. Для такого перехода необходимо подать логическую единицу на вход $D$, а затем кратковременно подать сигнал синхронизации – логическую единицу на вход $C$.

В момент, когда на входах $C$ и $D$ будут присутствовать единицы, в точке $S$ установится значение «0», тогда как в точке $R$ по-прежнему будет «1». Это приведёт к тому, что на выходе $Q$ появится «1». Эта единица тут же поступит на вход *правого нижнего* элемента «2И-НЕ», и в сочетании с единицей на входе $R$ установит на выходе $\overline{Q}$ значение «0». Этот ноль поступит на вход *правого верхнего* элемента «2И-НЕ» и зафиксирует новое состояние выходов: «$Q = 1$, $\overline{Q} = 0$». Далее, при снятии единицы со входа $C$ состояние триггера не изменится.

Чтобы переключить триггер обратно в состояние «$Q = 0$, $\overline{Q} = 1$», необходимо подать логический ноль на вход $D$, а затем сигнал синхронизации.

Получается, что триггер «сохраняет» на выходе $Q$ то значение, которое было на входе $D$ в момент поступления сигнала синхронизации («1») на вход $C$.

## Сколько будет 1+1?

Давайте теперь при помощи рассмотренных выше базовых логических элементов попытаемся реализовать операцию сложения двух одноразрядных двоичных чисел, следуя известному правилу (рис. 8).

| A | B | A+B *(в двоичной системе)* | A+B *(в десятичной системе)* |
|---|---|---|---|
| 0 | 0 | 0 + 0 = 0 | 0 |
| 0 | 1 | 0 + 1 = 1 | 1 |
| 1 | 0 | 1 + 0 = 1 | 1 |
| 1 | 1 | 1 + 1 = 10 | 2 |

Рис. 8

Хотелось бы для этой операции использовать элемент «2ИЛИ», реализующий как раз логическое сложение, но «картину портит» случай сложения двух единиц. На выходе элемента «2ИЛИ» появится логическая единица, тогда как должен появиться ноль. То есть надо построить логическую схему, реализующую операцию «исключающего ИЛИ» (XOR, неравнозначность). Такая логическая операция может быть описана формулой:

$$(A \vee B) \wedge \overline{(A \wedge B)} = A \text{ XOR } B.$$

| A | B | $A \vee B$ | $A \wedge B$ | $\overline{(A \wedge B)}$ | $(A \vee B) \wedge \overline{(A \wedge B)} = A \text{ XOR } B$ |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 |

Рис. 9

Это не единственная формула, которой может быть описана операция исключающего ИЛИ. Однако именно данная формула позволяет использовать минимум элементов для построения логической схемы (минимизированной схемы) (рис. 10).

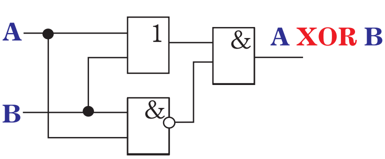



Рис. 10

Операция исключающего ИЛИ в компьютерной логике применяется достаточно часто, поэтому имеет собственное обозначение на логических схемах. Такой элемент называется *«четвертьсумматором»* (рис. 11).

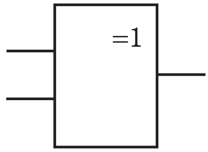



Рис. 11

Однако данный элемент не позволяет нам отличить ситуацию, когда на входах присутствуют две единицы, от ситуации двух нулей. А для *арифметического* сложения это необходимо! То есть если на оба входа элемента подать единицы, должен формироваться отдельный специальный сигнал о переполнении, который как раз и позволит отличить случай сложения двух нулей от случая сложения двух единиц. Получается, что логическая схема арифметического сложения двух одноразрядных двоичных чисел должна быть устроена иначе: на выходе должен присутствовать не один, а два сигнала, что будет соответствовать двухзначному двоичному числу. Или, говоря иначе, значение логического сложения по модулю 2 (выход $S$) и сигнал переполнения – переноса единицы в следующий разряд (выход $PO$). Возможные варианты таких логических схем представлены на рисунке 12.

$0 + 0 = 00_2$

$0 + 1 = 01_2$

$1 + 1 = 01_2$

$1 + 1 = 10_2$

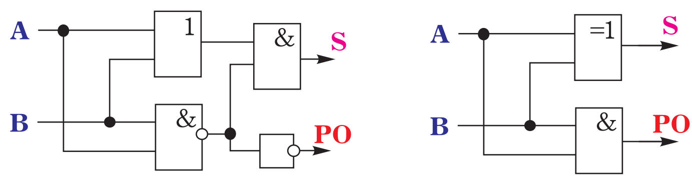



Рис. 12

Это устройство называется *«полусумматор»* и также имеет своё обозначение на схемах (рис. 13).

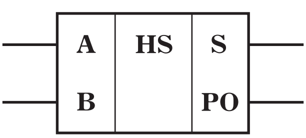



Рис. 13

Для сложения многоразрядных двоичных чисел используются *полные сумматоры*, в которых предусмотрен контроль перехода единицы в старший разряд. У такого сумматора три входа и два выхода (рис. 14).

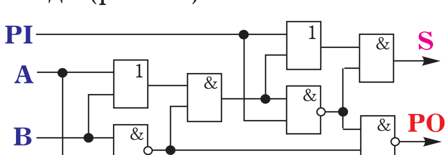



Рис. 14


На входы $A$ и $B$, как и прежде, подаются складываемые ариметически сигналы, а на вход *PI* – сигнал переноса из предыдущего разряда. Как вы можете заметить по рисунку, полный сумматор состоит из двух полусумматоров. Первый складывает сигналы *A* и *B*, а второй складывает полученный результат с сигналом *PI*. Назначение выходов такое же, как у полусумматора.

Работу данного устройства можно проиллюстрировать следующей таблицей (рис. 15).

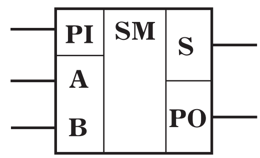
| *Входы* | | | *Выходы* | | *Обозначение полных сумматоров на схемах* |
|---|---|---|---|---|---|
| PI | A | B | PO | S |  |
| 0 | 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 0 | 1 | |
| 0 | 1 | 0 | 0 | 1 | |
| 0 | 1 | 1 | 1 | 0 | |
| 1 | 0 | 0 | 0 | 1 | |
| 1 | 0 | 1 | 1 | 0 | |
| 1 | 1 | 0 | 1 | 0 | |
| 1 | 1 | 1 | 1 | 1 | |



Рис. 15

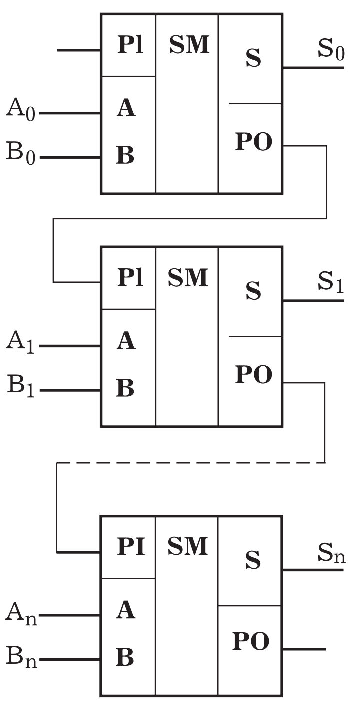



Рис. 16

Для сложения многоразрядных двоичных чисел сумматоры включаются таким образом, что выход *PO* предыдущего сумматора соединяется со входом *PI* следующего, образуя «цепочку» (рис. 16). (Первым элементом может быть также полусумматор.)

В этой статье мы рассмотрели лишь одно арифметическое действие. Возможно, у читателя появилась надежда: «железная» логика обладает-таки математическими способностями – главное всё правильно подключить! С другой стороны, сложение – это самая простая операция. А как у компьютера обстоят дела, например, с вычитанием? Да ещё в двоичной системе! Скажу вам по секрету: вы скоро узнаете, как ещё могут работать рассмотренные здесь схемы сложения и до чего довела компьютер человеческая смекалка. Продолжение следует...