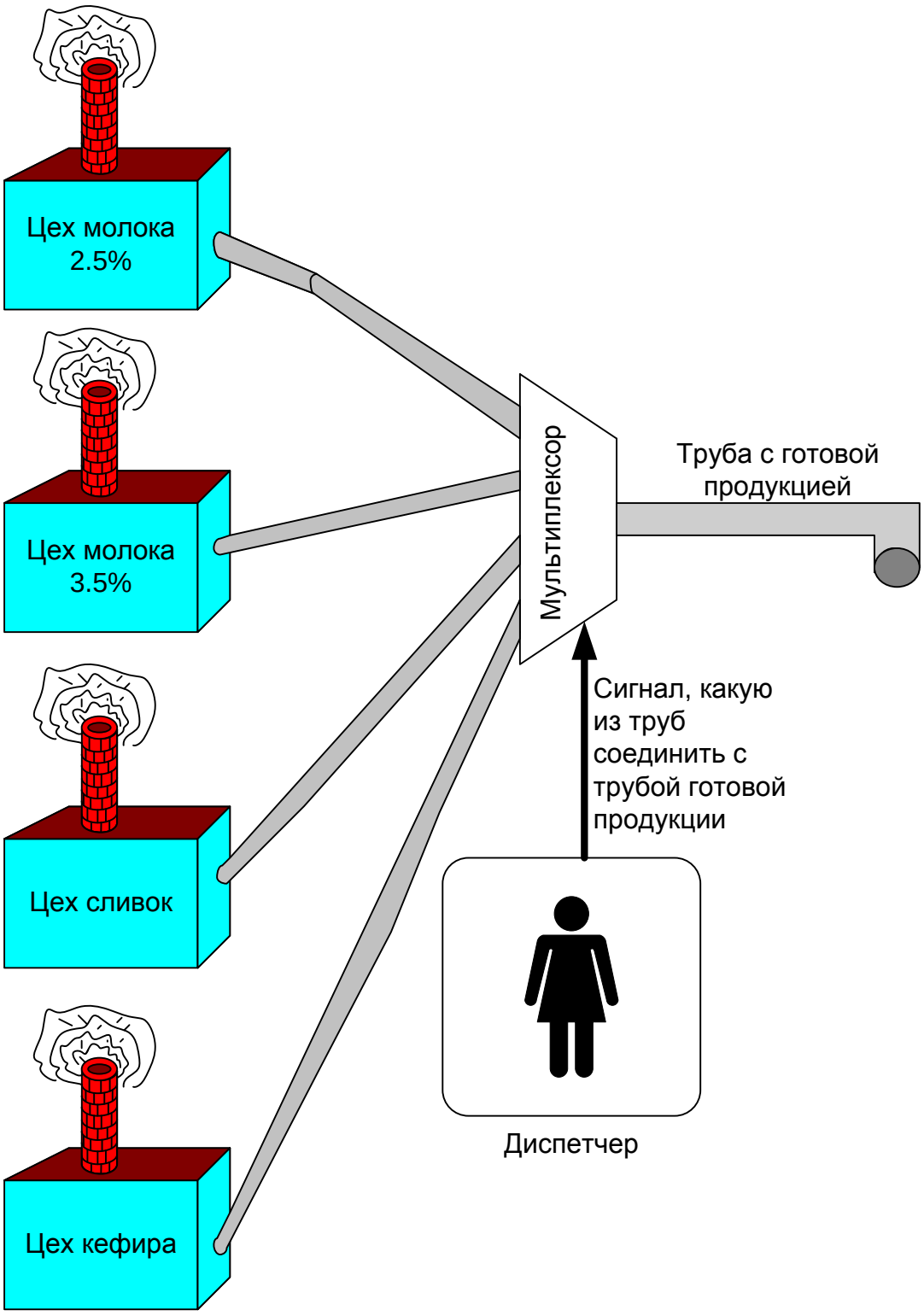
Общие сведения

Ну что ж. Мы медленно, но верно движемся к концу цикла статей об азах цифровой техники. Ещё чуть-чуть, и мы все эти азы изучим и будем готовы к дальнейшим свершениям. Давайте оглянемся на то, что мы уже знаем. Мы знаем о двух типах выходов (Открытый Коллектор и двухтактный), мы знаем, что выходы нельзя соединять друг с другом (если это не открытый коллектор, разумеется). Если хотим что-то соединить, то надо воспользоваться логическими функциями. Мы уже знаем простейшие логические функции (И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ и т.д.). Мы научились из логических функций собирать очень полезный элемент – триггер. Триггер – это также простейший элемент, из которого можно делать более сложные элементы (счётчики, параллельные регистры, регистры сдвига). При всей своей сложности, любой из этих элементов распадается сначала на простейшие логические функции, а затем – на транзисторы, но нас это не волнует. Мы берём такой типовой блок, какой нужен нам. Логично предположить, что как триггеры собираются в регистры и счётчики, так же и простейшие логические функции собираются в более сложные типовые функции. Давайте же рассмотрим их.

Мультиплексоры

Давайте представим себе некий гипотетический молочный комбинат. Пусть у него есть одна выходная труба, к которой подъезжают грузовики (понятно, что в жизни всё не так, но давайте представим). А этот комбинат может выпускать молоко 2.5%, молоко 3.5%, сливки и кефир. Из каждого цеха выходит труба с готовой продукцией, но на выход с территории комбината идёт всего одна. И сидит тётечка диспетчер, которая подключает выходную трубу к трубе одного из цехов. И переключает она всё при помощи мультиплексора. Приехала машина «МОЛОКО 2.5%» - мультиплексор соединил соответствующую трубу, машину залили. Приехала машина «Сливки» - молочный цех отключили, подключили сливочный… Не спрашивайте меня, почему у нас с молокозавода продукцию увозят машины, такой хитрый завод. Если придумаете более наглядный пример (и нарисуете картинки) – вставим его. А пока считаем, что у нас всё увозят машинами, чтобы удобнее было трубы рисовать. Может там не трубы, а ленты транспортёра, по которым ящики с пакетами ездят, я не проверял. Главное – есть несколько входов, есть – управляющий сигнал, который говорит, какой вход сейчас идёт на выход. Остальные входы – не играют никакой роли.

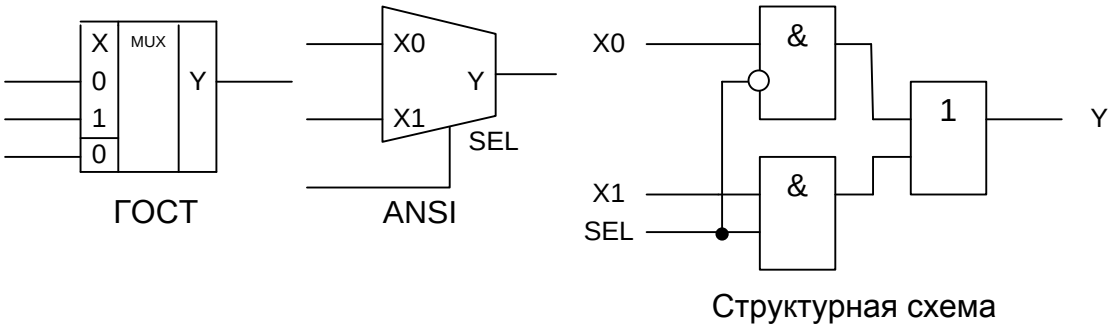


Собственно, в цифровой схемотехнике всё то же самое. Есть много входов, есть один выход, мультиплексор пропускает один выбранный вход на выход, остальные входы в это время никуда не идут.

Давайте посмотрим, как устроен мультиплексор. Начнём с простейшего – двухвходового. У него есть два входа и один управляющий сигнал. Таблица истинности такого мультиплексора (ура! Нет триггеров! Можно вернуться от временных диаграмм к старым добрым таблицам истинности!) выглядит так:

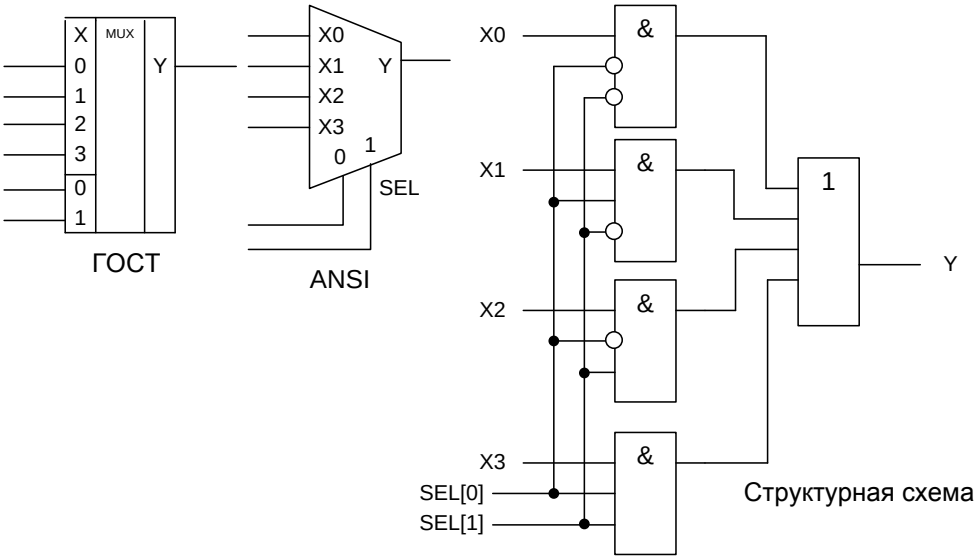
X0	X1	SEL	Y
0	X	0	0
1	X	0	1
X	0	1	0
X	1	1	1

То есть, когда SEL=0, выход Y повторяет значение входа X0, когда SEL=1, выход Y повторяет значение входа X1.



Увеличим число входов до четырёх. Теперь управляющих сигналов стало два – SEL[1] и SEL[0]. Правила выбора входа простое – двоичный код. Когда SEL[1..0] = 00, выбран вход 0, когда 01 – вход 1, 10 – вход 2, 11 – вход 3 (ещё не забыли правила счёта двоичных чисел? Если забыли – потренируйтесь, полезно поддерживать навык).

X0	X1	X2	X3	SEL[1]	SEL[0]	Y
0	X	X	X	0	0	0
1	X	X	X	0	0	1
X	0	X	X	0	1	0
X	1	X	X	0	1	1
X	X	0	X	1	0	0
X	X	1	X	1	0	1
X	X	X	0	1	1	0
X	X	X	1	1	1	1

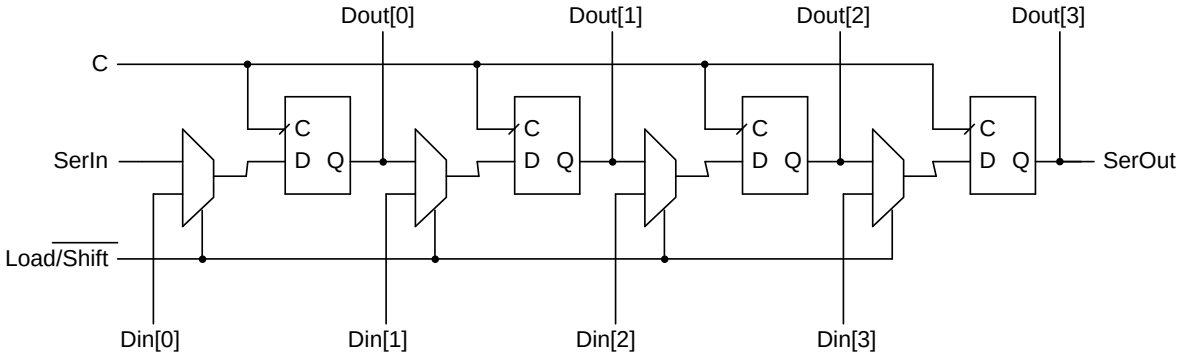


Аналогично строятся 8, 16, 32 и т.д. входовые мультиплексоры. Кстати, помните в статье про ПЛИС (кто не помнит – пробегите её по диагонали, она уже должна стать

понятнее) я писал, что все логические функции можно записать при помощи структуры И-ИЛИ-НЕ? Как видим, мультиплексоры строятся именно так, то есть, структура CPLD пока что весьма неплоха (что же до FPGA, то, как уже говорилось, там функция вообще не важна, там всё делается через загрузку таблиц истинности в ОЗУ, были бы таблицы истинности).

Практический пример использования мультиплексора

Регистр сдвига в том виде, в каком мы его рассмотрели, хорош для работы с последовательными данными (М-последовательность, бегущие огни и т.п) или преобразования последовательных данных в параллельные (приёмник COM порта). Однако, часто надо наоборот, преобразовать параллельные данные в последовательные. В стародавние времена, в качестве примера я привёл бы видеоадаптер. Когда комьютеры были слабенькими, один бит кодировал одну точку. Поэтому один байт кодировал восемь точек. И они выдавались в видеовыход как раз при помощи регистра сдвига – сначала байт из ОЗУ защёлкивался в триггерах, а затем – за 8 тактов его содержимое уходило на выход, задавая чёрный или белый цвет. К сожалению, сейчас такой пример устарел, сейчас одна точка кодируется даже не байтом, а тремя байтами (по байту на компоненту R,G и B). Но тогда возьмём простейший пример – передатчик COM порта. Там мы задаём байт, а затем – отправляем бит за битом, скажем, в DOME камеру. Так что было бы полезно сделать регистр сдвига с возможностью параллельной загрузки. И здесь нам на помощь приходит мультиплексор. Нарисуем регистр сдвига в таком виде:

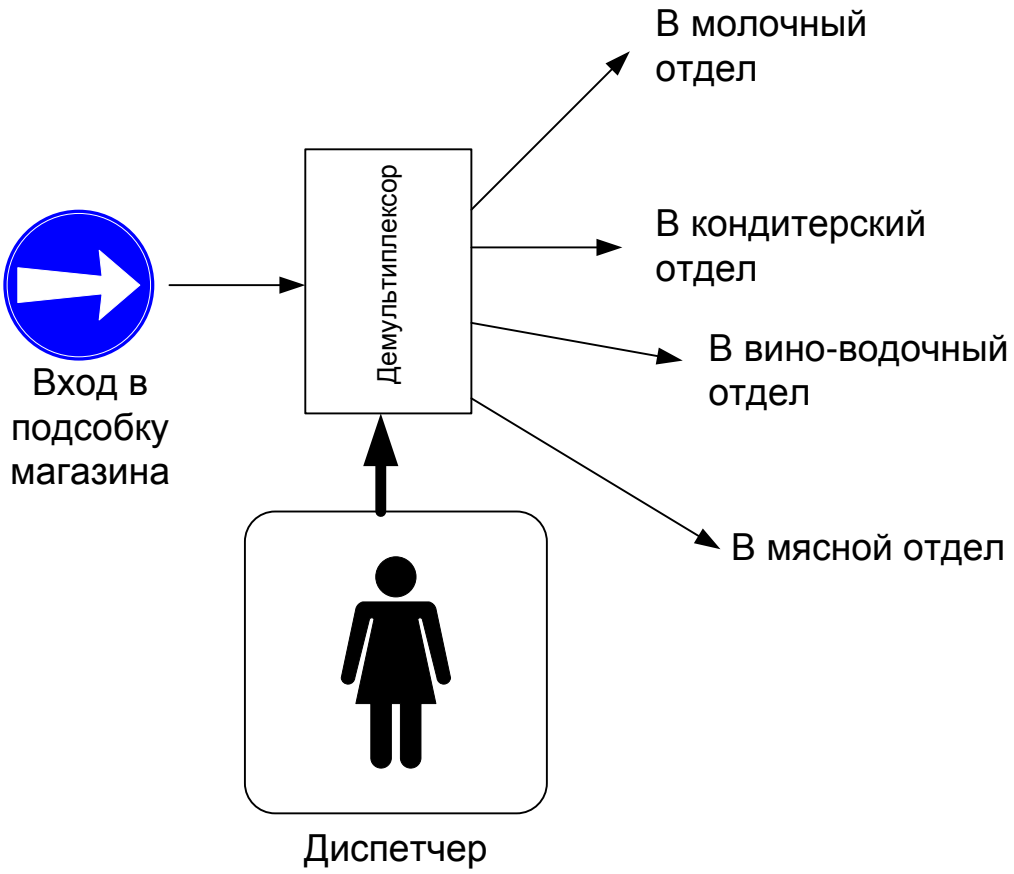


Теперь, если на управляющем входе мультиплексора единица, он находится в режиме LOAD – по фронту синала C, в триггерах защёлкнется значение, которое находится на входах Din[3..0]. Если же на управляющем входе мультиплексора ноль, он находится в режиме Shift, и по фронту сигнала C данные будут сдвигаться. При этом новые данные в регистр будут попадать из входа SerIn. То есть, такой регистр можно загружать как угодно, хоть параллельно через входы Din, хоть последовательно, через вход SerIn.

Демультиплексоры

Давайте теперь представим себе магазин. У него есть один вход, к которому подъезжают машины с товаром. А дальше товар расходится по отделам – в молочный отдел, в кондитерский, в вино-водочный, в мясной… То есть, теперь у нас есть один вход и несколько выходов, которыми опять же, управляет тётечка-диспетчер. Какая машина приехала, к тому отделу дорожка и открывается. Остальные дорожки в это время

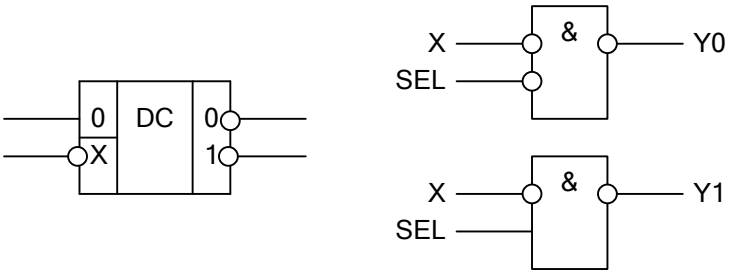
недоступны. Наверное, чтобы грузчики ничего по пути не спёрли. И делает она это при помощи демультиплексора.



Насчёт таблицы истинности демультиплексора, я долго думал, в каком виде её привести. В конце концов, решил привести её в несколько нелогичном виде, зато именно так работают стандартные демультиплексоры. Я сам до конца не понимаю, почему именно так, а не иначе, но видимо, были какие-то причины, которые потерялись в веках. Дело в том, что почему-то если выход пассивен, он принимает значение логической единицы. Ну, раз старые добрые микросхемы демультиплексоров делают именно так, на всякий случай, сделаю так же. Хотя бы для исключения путаницы…

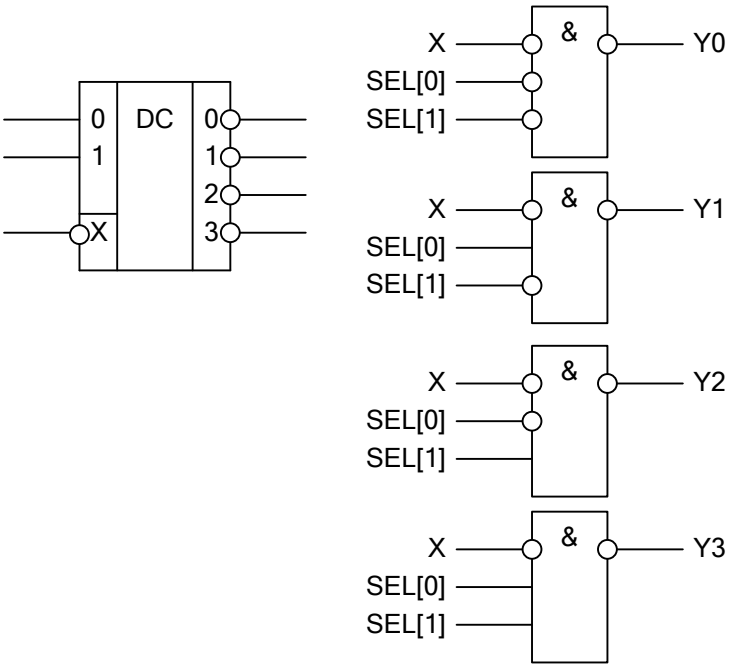
X	SEL	Y0	Y1
0	0	**0**	1
1	0	**1**	1
0	1	1	**0**
1	1	1	**1**

Графическое представление и структурная схема у демультиплексора, описанного по такой таблице истинности, выглядят так:



Ну, и закрепим материал на четырёхвыходном демультиплексоре:

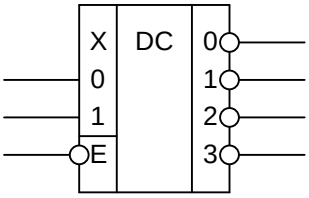
X	SEL[1]	SEL[0]	Y0	Y1	Y2	Y3
0	0	0	**0**	1	1	1
1	0	0	**1**	1	1	1
0	0	1	1	**0**	1	1
1	0	1	1	**1**	1	1
0	1	0	1	1	**0**	1
1	1	0	1	1	**1**	1
0	1	1	1	1	1	**0**
1	1	1	1	1	1	**1**



В чистом виде, мне демультиплексор использовать не доводилось, поэтому примеров не будет.

Двоичный дешифратор

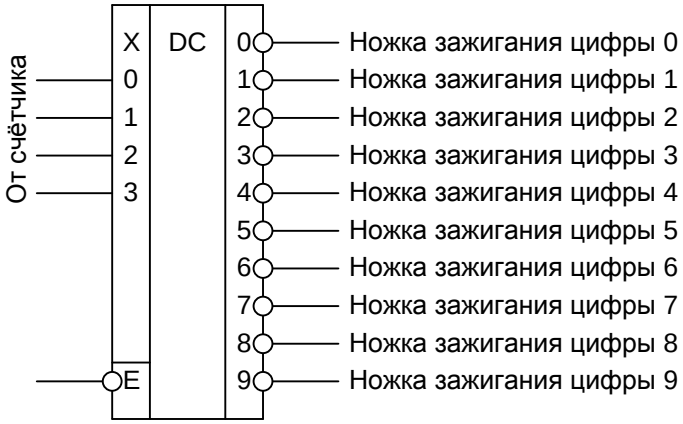
Однако, демультиплексор можно обозначить чуть иным способом. Давайте мы скажем, что входы SEL – это не входы управления, а главный вход в двоичном виде. А вход X – это вход Enable (или коротко – E). И посмотрим на таблицу истинности в этом случае:



E	X[1]	X[0]	Y0	Y1	Y2	Y3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	0	0	1	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Обратите внимание. Когда E=1, на выходах всегда 1, независимо от того, что на входе. А вот когда E=0, активизируется тот выход, двоичный код которого задан на входах X[1..0]. Если там 00, активизируется выход 0, если там 01 – выход 1, 10 – выход 2, 11 – выход 3. То есть мы производим дешифрацию двоичного кода в позиционный.

Давным давно, когда техника была ламповой, а семисегментные индикаторы ещё не были изобретены, были специальные лампы, у которых шло десять символов подряд. Каждый символ подавался на свою ножку. На какую ножку подан сигнал, та и светилась (подадим на все – получим кашу малашу, так как светиться будут все). Посему первые электронные часы декодировали цифры именно таким путём. Сначала стоял дешифратор из двоичного кода в позиционный, а позиционный код подавался на сигнал зажигания соответствующей цифры.



В 21-м веке про лампы говорить – смешно, но так как у меня скоро самолёт, про более жизненные применения дешифраторов мы поговорим в следующий раз, может даже посвятив этому целую статью.