Лекция 8. Проектирование комбинационных логических микросхем

Цель лекции – дать студентам четкое понимание принципов проектирования комбинационных логических микросхем и их применения в цифровых системах.

Введение

Комбинационными называются логические устройства, выходные сигналы которых однозначно определяются комбинацией входных сигналов в тот же момент времени.

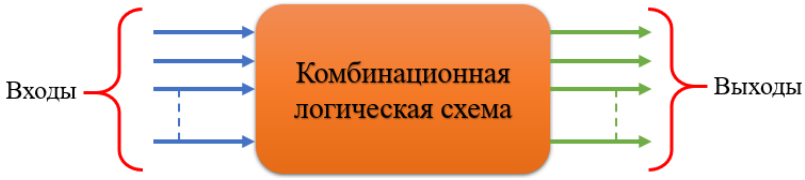

Рисунок 8.1. Общая схема комбинационных логических микросхем

Логические элементы И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ являются строительными блоками комбинационных логических схем. Примером комбинационной схемы является дешифратор, который преобразует двоичные кодовые данные, присутствующие на его входе, в ряд различных выходных линий, по одной за раз производя эквивалентный десятичный код на своем выходе. Комбинационные логические схемы могут быть очень простыми или очень сложными, и любая комбинационная схема может быть реализована только с использованием вентилей И-НЕ и ИЛИ-НЕ, поскольку они классифицируются как «универсальные» вентили.

Три основных способа задания функции комбинационной логической схемы: Булева алгебра, таблица истинности и логическая схема.

К распространенным комбинационным схемам, состоящим из отдельных логических вентилей, относятся мультиплексоры, демультиплексоры, шифраторы, дешифраторы, сумматоры, компараторы и т. д.

Сумматор

Наиболее распространенной арифметической операцией является суммирование. Кроме того, сумматор также часто является основным элементом, ограничивающим быстродействие схемы. Следовательно, к разработке и оптимизации сумматора необходимо относиться очень внимательно. Эту оптимизацию можно выполнить либо на уровне логических элементов, либо на уровне схемы.

Полный сумматор представляет собой логическую схему, которая выполняет операцию сложения трех двоичных цифр, а также генерирует перенос в следующий столбец сложения.

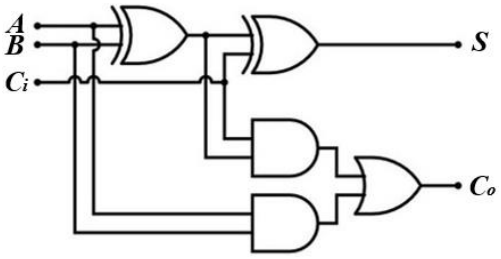

Рисунок 8.2. Реализация полного сумматора на логических элементах

Тогда перенос на вход представляет собой возможный перенос из разряда меньшей значимости, тогда как перенос на выход представляет собой перенос в разряд большей значимости. Двоичные сумматоры позволяют суммировать два двоичных числа и являются основным блоком процессора. В табл. 8.1 приведена таблица истинности двоичного полного сумматора, где A и B – входы сумматора, C_i – входной сигнал переноса, S – выходной сигнал суммы, C_o – выходной сигнал переноса.

Таблица 8.1. Таблица истинности полного сумматора

A	B	C_i	S	C_0	Состояние сигнала переноса
0	0	0	0	0	Удалить
0	0	1	1	0	Удалить
0	1	0	1	0	Передать
0	1	1	0	1	Передать
1	0	0	1	0	Передать
1	0	1	0	1	Передать
1	1	0	0	1	Сгенерировать / передать
1	1	1	1	1	Сгенерировать / передать

Булевы выражения для S и C_o представлены в уравнении (8.1):

$$S = A \oplus B \oplus C_i = A\bar{B}\bar{C_i} + \bar{A}B\bar{C_i} + \bar{A}\bar{B}C_i + ABC_i; \quad (8.1)$$
$$C_0 = AB + BC_i + AC_i.$$

С точки зрения реализации S и C_o часто полезно определить как функции некоторых промежуточных сигналов G («сгенерировать»), D («удалить») и P («передать»). G = 1 (D = 1) гарантирует генерацию (удаление) разряда переноса в C_o независимо от C_i, а P = 1 гарантирует передачу в C_o входного сигнала переноса. Выражения для этих сигналов можно получить, просто изучив таблицу истинности:

$$G = AB; D = \bar{A}\bar{B}; P = A \oplus B \quad (8.2)$$

Сигналы S и C_o мы можем переписать как функции P и G (или D):

$$C_0(G, P) = G + PC_i; \;\; S(G, P) = P \oplus C_i \quad (8.3)$$

Обратите внимание на то, что G и P зависят только от A и B и не зависят от C_i. Аналогичным образом мы можем вывести выражения для $S(D, P)$ и $C_o(D, P)$.

Один из способов реализации схемы полного сумматора заключается в трансляции логических уравнений (8.1) непосредственно в КМОП-схему. Число требуемых транзисторов можно уменьшить с помощью определенных логических манипуляций. Например, полезным является совместное использование некоторых логических элементов подсхемами генерации сигналов суммы и переноса, если это не замедляет генерацию сигнала переноса (как отмечалось ранее, это наиболее критичный путь). Пример подобного переупорядочения членов уравнения приведен ниже.

$$S = ABC_i + C_0(A + B + C_i); \quad (8.4)$$
$$C_0 = AB + BC_i + AC_i.$$

Легко проверить, что данная пара уравнений эквивалентна исходной. Схема соответствующего сумматора, реализованного на КМОП, показана на рис. 8.3. Она

требует 28 транзисторов. Помимо того, что данная схема имеет слишком большую площадь, она является очень медленной.

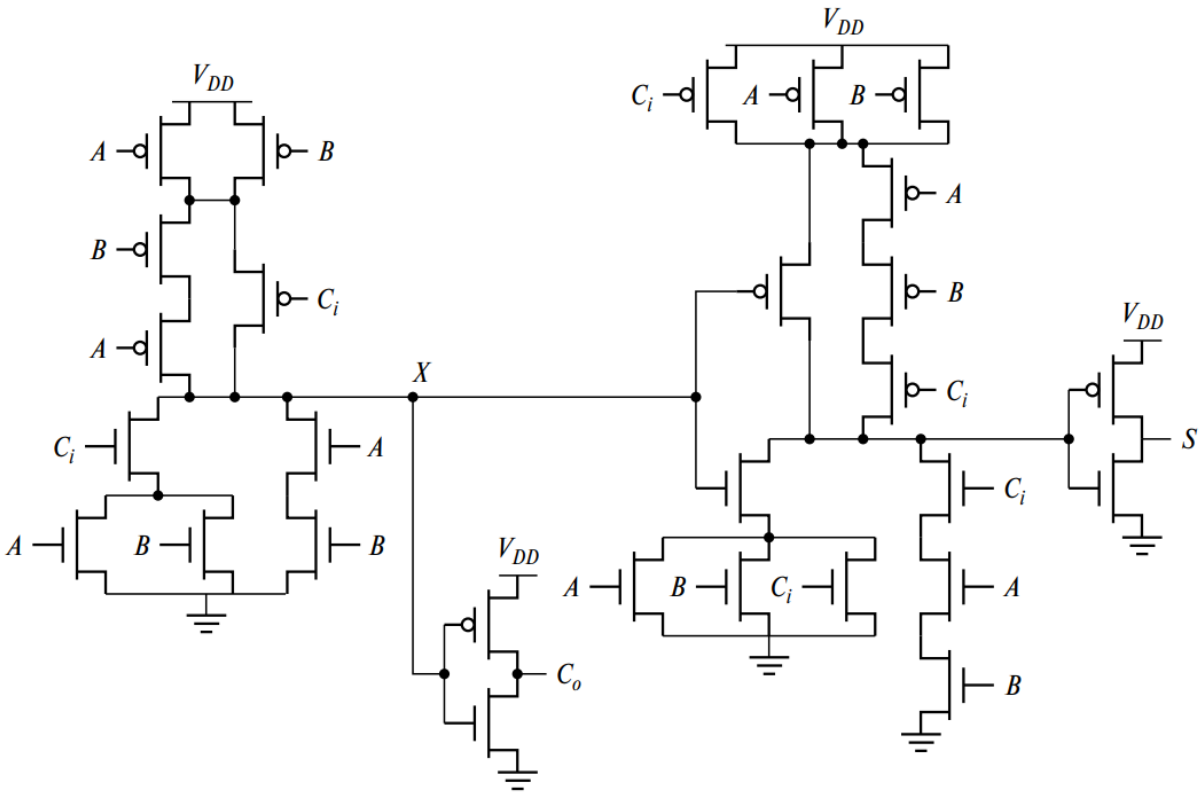

Рисунок 8.3. Реализация полного сумматора на КМОП-схемах

На рис. 8.4 показана улучшенная схема сумматора – так называемый зеркальный сумматор (mirror adder), действие которого основано на уравнении (8.3).

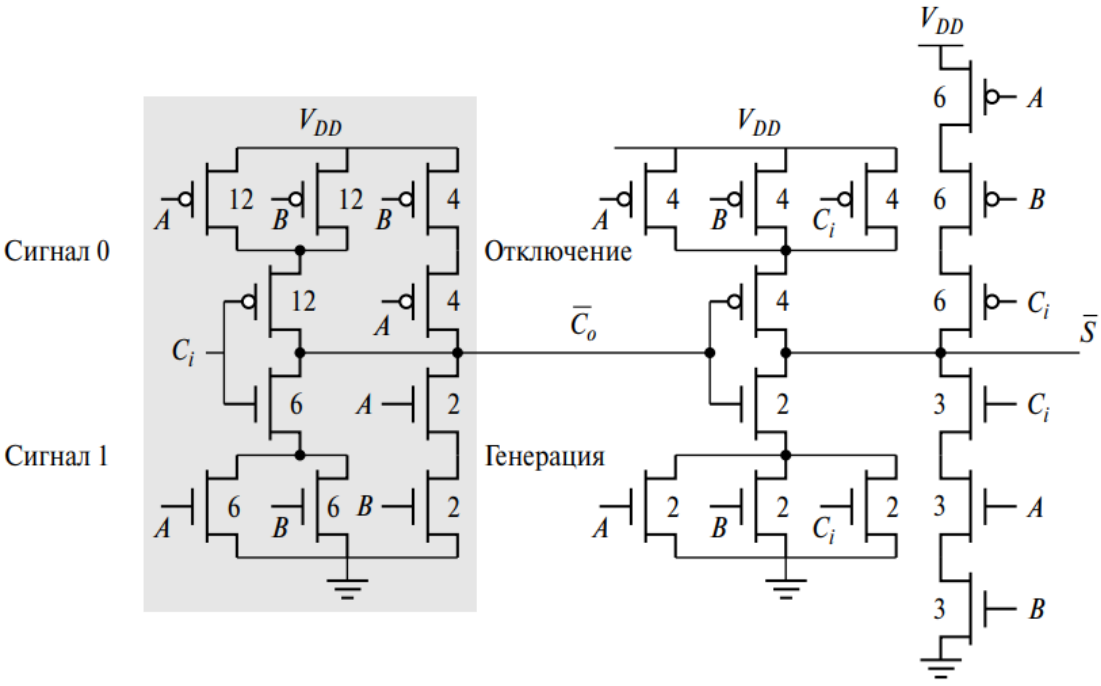

Рисунок 8.4. Зеркальный сумматор

Данная ячейка полного сумматора требует всего 24 транзистора. Цепочки n-МОП- и p-МОП-транзисторов являются полностью симметричными, что не мешает правильной работе из-за самодуальности функций суммирования и переноса. В результате схема генерации сигнала переноса содержит максимум два последовательно включенных транзистора.

Мультиплексор

Мультиплексор – это комбинационная логическая схема, предназначенная для переключения одной из нескольких входных линий на одну общую выходную линию путем подачи управляющего сигнала. Мультиплексоры работают как очень быстродействующие многопозиционные поворотные переключатели, соединяющие или управляющие несколькими входными линиями, называемыми «каналами», по одной за раз к выходу.

Мы можем построить простой мультиплексор 2x1 из базовых логических элементов И-НЕ, как показано на рис. 8.5.

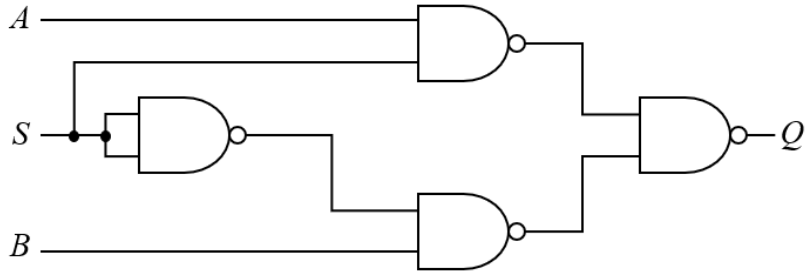

Рисунок 8.5. Реализация 2x1 мультиплексора на логических элементах и его таблица истинности

Таблица 8.2. Таблица истинности двухвходового мультиплексора

S	***A***	***B***	***Q***
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Вход A этой простой схемы 2x1 мультиплексора, построенной на стандартных вентилях И-НЕ, управляет тем, какой вход (*A* или *B*) передается на выход *Q*. Из приведенной выше таблицы истинности видно, что когда вход выбора данных *S* имеет НИЗКИЙ уровень (логический 0), вход *A* передает свои данные через схему мультиплексора на вентилях И-НЕ на выход, в то время как вход *B* блокируется. Когда выбор данных *S* имеет ВЫСОКИЙ уровень (логическая 1), происходит обратное, и теперь вход *B* передает данные на выход *Q*, в то время как вход *A* заблокирован. Таким образом, применяя либо логический «0», либо логическую «1» к *S*, мы можем выбрать соответствующий вход, *A* или *B*. Логическое выражение для 2x1 мультиплексора можно написать в виде:

$$Q = \bar{S}A + AB \tag{8.6}$$

Мы можем увеличить количество выбираемых входов данных, просто следуя той же процедуре, и более крупные схемы мультиплексора могут быть реализованы с использованием меньших 2x1 мультиплексоров в качестве их базовых строительных блоков. Таким образом, для мультиплексора с 4 входами нам потребуются две линии выбора данных, поскольку 2^2 входа представляют 4 линии управления данными, что дает схему с четырьмя входами, *A, B, C, D*, и двумя линиями выбора данных S_1 и S_2, как показано на рис. 8.6.

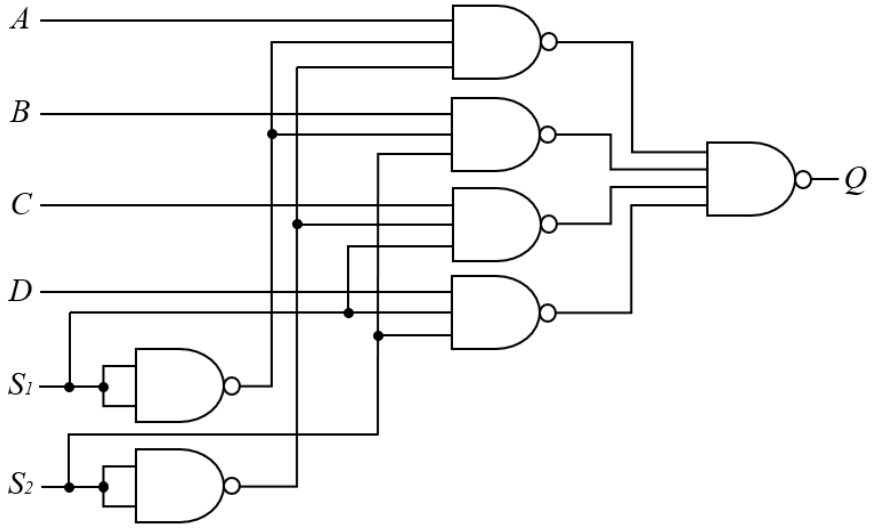

Рисунок 8.6. Реализация 4x1 мультиплексора на логических элементах и его таблица истинности

Таблица 8.3. Таблица истинности 4x1 мультиплексора

S_2	S_1	D	C	B	A	Q
0	0	x	x	x	1	1
0	1	x	x	1	x	1
1	0	x	1	x	x	1
1	1	1	x	x	x	1

Булево выражение для 4x1 мультиплексора, представленного выше, с входами *A, B, C, D* и линиями выбора данных S_1, S_2 задается как:

$$Q = \overline{S_1 S_2}A + S_1\bar{S_2}B + \bar{S_1}S_2C + S_1S_2D \tag{8.7}$$

В этом примере в любой момент времени замкнут только один из четырех аналоговых переключателей, подключая только одну из входных линий к единственному выходу на Q. То, какой переключатель замкнут, зависит от кода адресации входных линий на линиях S_1 и S_2. Символ, используемый в логических схемах для обозначения мультиплексора, выглядит как на рис. 8.7.

Можно сделать простые схемы мультиплексора из стандартных вентилей И-НЕ, как мы видели выше, но обычно мультиплексоры доступны в виде стандартных корпусов интегральных схем, таких как обычный мультиплексор на ТТЛ-логике 74LS151 с 8 входами на 1 выход или ТТЛ-мультиплексор 74LS153 с двумя 4 входами на 1 выход.

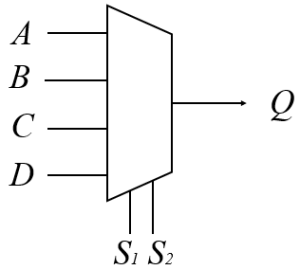

Рисунок 8.7. Условное обозначение мультиплексора

Цифровой компаратор

Цифровой компаратор – еще одна очень полезная комбинационная логическая схема, используемая для сравнения значений двух двоичных цифр. Двоичный или цифровой компаратор может быть построен с использованием стандартных вентилей И, ИЛИ-НЕ и НЕ для сравнения цифровых сигналов на входах, и формирования выходного сигнала в зависимости от состояния этих входов.

Например, наряду с возможностью складывать и вычитать двоичные числа, нам необходимо уметь сравнивать их и определять, является ли значение на входе *A* большим, меньшим или равным значению на входе *B* и т. д. Цифровой компаратор выполняет эту задачу с помощью нескольких логических вентилей, работающих по принципам булевой алгебры.

Целью цифрового компаратора является сравнение набора переменных или неизвестных чисел, например *A* (A_1, A_2, A_3, ... A_n и т. д.), с константой или неизвестным значением, например *B* (B_1, B_2, B_3, ... B_n и т. д.), и создание выходного условия или флажка в зависимости от результата сравнения. Компаратор двух 1-битных входов (*A* и *B*) при сравнении друг с другом выдаст следующие три выходных условия: $A > B$, $A = B$, $A < B$. Это означает: *A* больше *B*, *A* равно *B* или *A* меньше *B*. Это полезно, если мы хотим сравнить две переменные и хотим выдать выходной сигнал при достижении любого из трех вышеуказанных условий. Например, выдать выходной сигнал счетчика при достижении определенного числа. Рассмотрим простой 1-битный компаратор ниже.

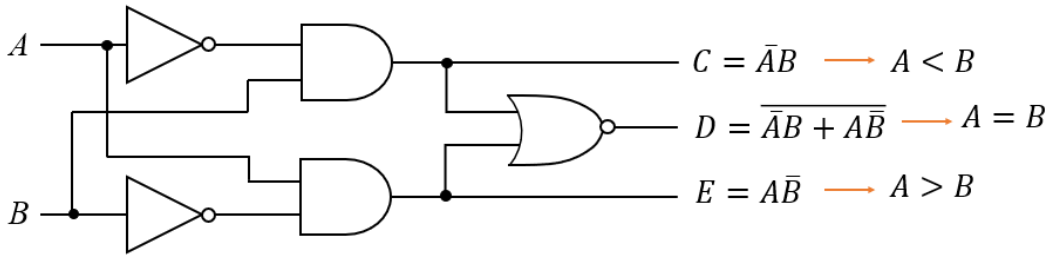

Рисунок 8.8. 1-bit Digital Comparator Circuit

Таблица 8.4. Таблица истинности цифрового компаратора

Входы		**Выходы**		
B	***A***	***A* > *B***	***A* = *B***	***A* < *B***
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

Некоторые коммерчески доступные цифровые компараторы, такие как 4-битный ТТЛ-компаратор 74LS85 или КМОП 4063, имеют дополнительные входные терминалы, которые позволяют «каскадировать» большее количество отдельных компараторов для сравнения слов размером более 4 бит с получением компараторов из «*n*» бит.

Шифратор

Шифратор (кодер) принимает все свои входные данные по одному за раз, а затем преобразует их в один закодированный выход. Таким образом, мы можем сказать, что шифратор – это многовходовая комбинационная логическая схема, которая преобразует данные логического уровня «1» на своих входах в эквивалентный двоичный код на своем выходе. Обычно шифраторы выдают на выходе 2-битные, 3-битные или 4-битные коды в зависимости от количества входных данных. Двоичный шифратор «*n*-бит» имеет 2^n входных линий и *n*-битных выходных линий с общими типами, которые включают конфигурации линий 4x2, 8x3 и 16x4. Выходные линии шифратора генерируют двоичный эквивалент входной линии, значение которой равно «1», и могут кодировать либо десятичный, либо шестнадцатеричный входной шаблон в обычно двоичный или двоично-десятичный (BCD – binary-coded decimal) выходной код.

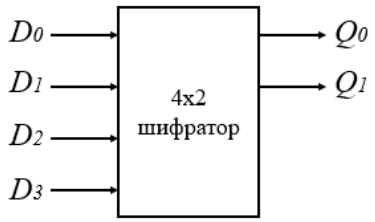

Рисунок 8.9. 4x2 двоичный шифратор

Таблица 8.5. Таблица истинности 4x2 шифратора

D_3	D_2	D_1	D_0	Q_1	Q_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1
0	0	0	0	x	x

Одним из главных недостатков стандартных шифраторов является то, что они могут генерировать неправильный выходной код, когда присутствует более одного входа на логическом уровне «1». Например, если мы зададим на входы D_1 и D_2 «1», то результирующий выход не будет ни «01», ни «10», а «11», что является выходным двоичным числом, которое отличается от фактического присутствующего входа. Кроме того, выходной код всех логических «0» может быть сгенерирован, когда все его входы будут «0» или когда вход D_0 равен единице.

Один простой способ преодолеть эту проблему – «приоритезировать» уровень каждого входного контакта. Таким образом, если одновременно имеется более одного входа со значениями «1», фактический выходной код будет соответствовать только входу с наивысшим назначенным приоритетом. Тогда этот тип шифратора обычно называют приоритетным шифратором.

Приоритетный шифратор решает проблемы, упомянутые выше, назначая каждому входу уровень приоритета. Выход приоритетного шифратора соответствует текущему активному входу, имеющему наивысший приоритет. Таким образом, когда присутствует

вход с более высоким приоритетом, все остальные входы с более низким приоритетом будут игнорироваться.

Приоритетный шифратор существует во многих различных формах, ниже приведен пример приоритетного шифратора с 8 входами и его таблица истинности.

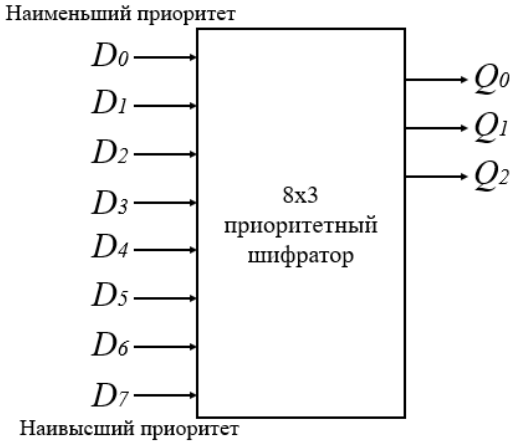

Рисунок 8.10. 8x3 двоичный шифратор

Таблица 8.6. Таблица истинности 8x3 шифратора

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	x	0	0	1
0	0	0	0	0	1	x	x	0	1	0
0	0	0	0	1	x	x	x	0	1	1
0	0	0	1	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	1	0	1
0	1	x	x	x	x	x	x	1	1	0
1	x	x	x	x	x	x	x	1	1	1

Приоритетные кодеры выводят сначала входной сигнал наивысшего порядка, например, если входные линии D_2, D_3 и D_5 применяются одновременно, выходной код будет для входа D_5 (101), поскольку он имеет наивысший порядок из трех входов. После удаления входа D_5 следующим наивысшим выходным кодом будет вход D_3 (011) и т. д.

В таблице знак «x» означает, что он может быть 0 или 1, но его значение будет игнорироваться (не имеет значения). Из этой таблицы истинности булево выражение для шифратора выше с входами данных D_0-D_7 и выходами Q_0, Q_1, Q_2 задается как:

$$Q_0 = \sum(\overline{D}_6(\overline{D}_4\overline{D}_2D_1 + \overline{D}_4D_3 + D_5) + D_7);$$
$$Q_1 = \sum(\overline{D}_5\overline{D}_4(D_2 + D_3) + D_6 + D_7); \quad (8.8.)$$
$$Q_2 = \sum(D_4 + D_5 + D_6 + D_7).$$

Шифратор можно построить из приведенного выше выражения, используя вентили ИЛИ как показано на рис. 8.11.

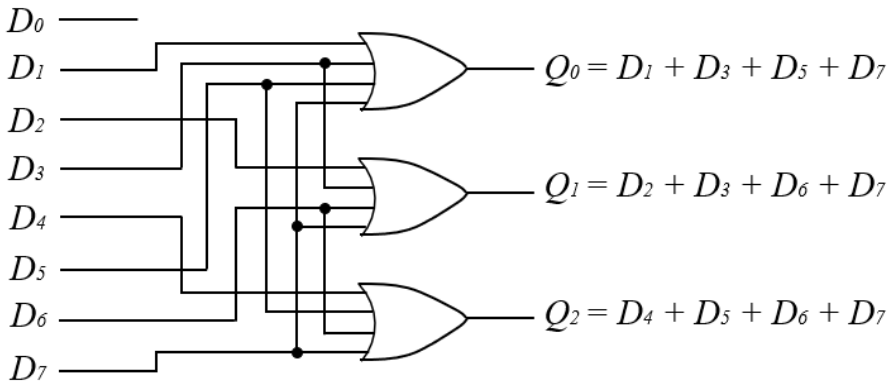

Рисунок 8.11. Цифровой шифратор с использованием логического элемента ИЛИ

Приоритетные шифраторы можно использовать для сокращения количества проводов, необходимых в конкретных схемах или приложениях, имеющих несколько входов.

Контрольные вопросы:

1. Что такое комбинационные логические схемы и как они работают?
2. Что такое полный сумматор и какие функции он выполняет?
3. Каковы булевы выражения для выхода суммы и сигнала переноса полного сумматора?
4. Что такое мультиплексор и какова его основная функция?
5. Каковы булевы выражения для 2x1 и 4x1 мультиплексоров?
6. Каковы функции и применения цифрового компаратора?
7. Что такое шифратор и как он преобразует входные данные?
8. Как можно построить шифратор из логических элементов?