ЛОГИЧЕСКИЕ ЭЛЕМЕНТЫ

Цифровые системы – это системы, способные выполнять различные операции с бинарными переменными. Логические вентили (logic gates) или логические элементы – это простейшие цифровые схемы, получающие один или более двоичных сигналов на входе и производящие новый двоичный сигнал на выходе.

При графическом изображении логических вентилей для обозначения одного или нескольких входных сигналов и выходного сигнала используются специальные символы.

Если смотреть на изображение логического элемента, то входные сигналы обычно размещаются слева (или сверху), а выходные сигналы – справа (или снизу). Разработчики цифровых систем обычно используют первые буквы латинского алфавита для обозначения входных сигналов и латинскую букву Y для обозначения выходного сигнала.

Взаимосвязь между входными сигналами и выходным сигналом логического вентиля может быть описана с помощью таблицы истинности (truth table) или уравнением Булевой логики.

Слева в таблице истинности представлены значения входных сигналов, а справа – значение соответствующего выходного сигнала.

Каждая строка в такой таблице соответствует одной из возможных комбинаций входных сигналов. Уравнение Булевой логики – это математическое выражение, описывающее логический элемент с помощью двоичных переменных.

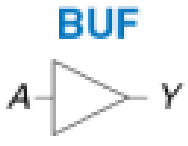
Логический вентиль НЕ (Логический элемент НЕ)

$Y = \overline{A}$

A	Y
0	1
1	0

Логический элемент НЕ (NOT gate) имеет один вход A и один выход Y, как показано на Рис. ниже. Причем выходной сигнал Y – это сигнал, обратный входному сигналу A, или, как еще говорят, инвертированный A (inversed A). Если сигнал на входе A – это ЛОЖЬ, то сигнал на выходе Y будет ИСТИНА. Таблица истинности и уравнение Булевой логики для элемента представлено ниже. В уравнении Булевой логики линия над обозначением сигнала читается как «не», то есть математическое выражение $Y = \overline{A}$ произносится как «Y равняется не A». Именно поэтому логический элемент НЕ также называют инвертором (inverter). Для обозначения логического вентиля НЕ используют и другие способы записи, включая: Y = A', Y = ¬A, Y = !A и Y = ~A в научной и технической литературе.

Буфер, логический буфер, цифровой буфер, неинвертирующий буфер



$Y = A$

A	Y
0	0
1	1

Другим примером логического вентиля - элемента с одним входом является буфер (buffer). Буфер просто копирует входной сигнал на выход. Если рассматривать буфер как часть логической схемы, то такой элемент ничем не отличается от простого провода и может показаться бесполезным. Вместе с тем, на аналоговом уровне буфер может обеспечить характеристики, необходимые для нормального функционирования разрабатываемого устройства.

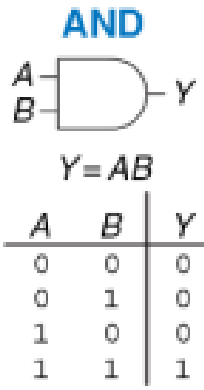
Буфер, например, необходим для передачи большого тока электродвигателю или для быстрой передачи сигнала сразу нескольким логическим элементам.

В логических схемах буфер обозначается треугольником. Кружок на выходе логического элемента, в англоязычной литературе часто называемый пузырем (bubble), указывает на инверсию сигнала, как, например, показано на Рис. с элементом НЕ.

Может после следующего абзаца станет чуть понятнее:

Этот тип буфера не выполняет инверсии или возможности принятия решений. Цифровой буфер с одним входом отличается от инвертора. Он не инвертирует и не изменяет свой входной сигнал каким-либо образом. Он считывает вход и выводит значение. Обычно входная сторона считывает либо ВЫСОКИЙ, либо НИЗКИЙ вход и выводит ВЫСОКОЕ или НИЗКОЕ значение соответственно. Посылает ли выходной терминал сигнал ВЫСОКОГО или НИЗКОГО уровня, определяется его входным значением. Выходное значение будет высоким, если и только если входное значение высокое. Другими словами, Q будет высоким, если и только если A имеет ВЫСОКИЙ уровень.

Логический элемент И

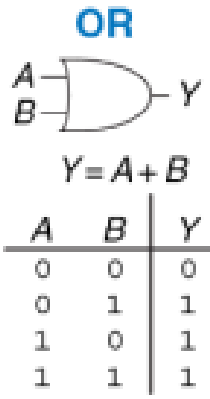


A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Логический элемент И (AND gate) выдает значение ИСТИНА на выход Y исключительно только если оба входных сигнала A и B имеют значение ИСТИНА. В противном случае выходной сигнал Y имеет значение ЛОЖЬ. Пусть входные сигналы перечислены в порядке 00, 01, 10, 11, как в случае подсчета в двоичной системе счисления (0,1,2,3).

Уравнение Булевой логики для логического элемента И может быть записано несколькими способами: Y = A • B, Y = AB, или Y = A ∩ B. Символ ∩ читается как «пересечение». Выражение Y = AB звучит как «Y равно A и B».

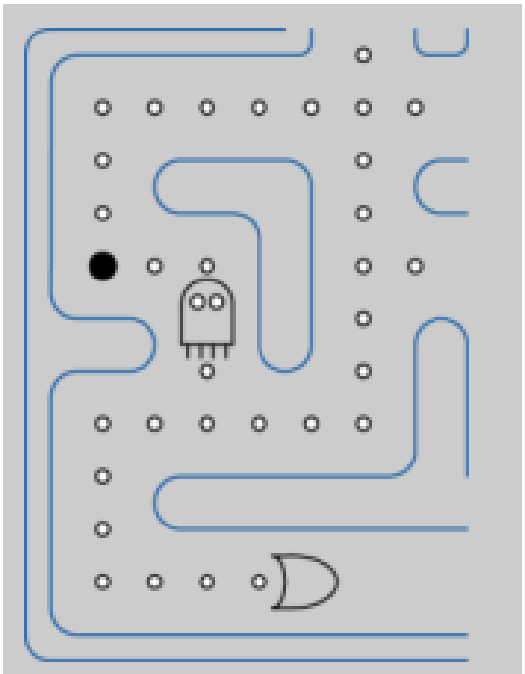
Логический элемент ИЛИ



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Логический вентиль ИЛИ (OR gate) выдаёт значение ИСТИНА на выход Y, если хотя бы один из двух входных сигналов A или B имеет значение ИСТИНА. Уравнение Булевой логики для логического элемента ИЛИ записывается как Y = A + B или Y = A ∪ B. Символ ∪ читается как «объединение». Разработчики цифровых систем обычно пользуются простым символом +. Математическое выражение Y = A + B звучит «Y равно A или B».

Забавный способ запомнить, как обозначается элемент ИЛИ на логических схемах, заключается в том, что графический символ ИЛИ напоминает главный персонаж компьютерной игры Pacman. Причем, широко раскрытая пасть “голодного” ИЛИ находится со стороны входных сигналов и готова проглотить все сигналы ИСТИНА, которые только может найти!



Другие логические элементы с двумя входными сигналами

$Y = A \oplus B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

$Y = \overline{AB}$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

$Y = \overline{A+B}$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

На Рис. показаны другие широко распространенные логические элементы с двумя входными сигналами. Добавление кружка на выходе любого логического вентиля превращает этот вентиль в ему противоположный – то есть инвертирует его.

Таким образом, например, из вентиля И получается вентиль **И-НЕ (NAND** gate). Значение выходного сигнала Y вентиля И-НЕ будет ИСТИНА до тех пор, пока оба входных сигнала A и B не примут значение ИСТИНА.

Точно так же из логического вентиля ИЛИ получается вентиль **ИЛИ-НЕ (NOR** gate). Его выходной сигнал Y будет ИСТИНА в том случае, если ни один из входных сигналов, ни A ни B, не имеет значение ИСТИНА.

Исключающее ИЛИ с количеством входов равным N (**N-input XOR** gate) иногда еще называют **элементом контроля по чётности (parity gate)**. Такой вентиль выдает на выход сигнал ИСТИНА, если нечетное количество входных сигналов имеет значение ИСТИНА.

Элемент **исключающее ИЛИ-НЕ (XNOR)**

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Показаны обозначение и **булевское (булевое, булево) уравнение** для вентиля исключающее ИЛИ-НЕ (XNOR) с двумя входами, который выполняет инверсию исключающего ИЛИ. Выход исключающего ИЛИ-НЕ есть ИСТИНА, если оба входа имеют значение ЛОЖЬ или оба входа имеют значение ИСТИНА.

Вентиль исключающее ИЛИ-НЕ с двумя входами иногда называют **вентилем равенства**, так как его выход есть ИСТИНА, когда входы совпадают.

Логические элементы с количеством входов больше двух

$Y = \overline{A+B+C}$

Многие Булевы функции, а значит, и логические элементы, необходимые для их реализации, оперируют тремя и более входными сигналами. Наиболее распространенные из таких вентилей - элементов – **это И, ИЛИ, Исключающее ИЛИ, И-НЕ, ИЛИ-НЕ и Исключающее ИЛИ-НЕ.**

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

Пример: **вентиль ИЛИ-НЕ с тремя входами.** На Рис. показаны обозначение и булевское уравнение для вентиля ИЛИ-НЕ с тремя входами. Выход есть ИСТИНА только, если нет ни одного входа со значением ИСТИНА.

БУЛЕВЫ УРАВНЕНИЯ

Булевы уравнения (Y = …) используют переменные, имеющие значение ИСТИНА или ЛОЖЬ, поэтому они идеально подходят **для описания цифровой логики**.

Терминология:

Дополнение (complement) переменной A – это ее отрицание $\overline{A}$.

Переменная или ее дополнение **называются литералом**. Например, A, $\overline{A}$, B и $\overline{B}$ – литералы.

Мы будем называть **A прямой формой переменной**, а $\overline{A}$ **– комплементарной формой**; «прямая форма» не подразумевает, что значение A равно ИСТИНЕ, а говорит лишь о том, что у A нет черты сверху.

Операция «И» над одним или несколькими литералами называется **конъюнкцией**, **произведени**ем (product) **или импликантой**. $\overline{A}$ B, A $\overline{B}$ $\overline{C}$ и B являются импликантами для функции трех переменных.

Минтерм (minterm, элементарная конъюнктивная форма) – это произведение, включающее все входы функции. A$\overline{B}$ $\overline{C}$ – это минтерм для функции трех переменных A,B и C, а $\overline{A}$B – не минтерм, поскольку он не включает в себя C (это безусловно для ситуации, когда изначально три входных переменных заявлено).

Аналогично, **операция «ИЛИ»** над одним или более литералами называется **дизъюнкцией или суммой**.

Макстерм (maxterm, элементарная дизъюнктивная форма) – это сумма всех входов функции. A+ $\overline{B}$ + C является макстермом функции трех переменных A, B и C.

Порядок операций важен при анализе булевых уравнений. Означает ли Y = A + BC, что Y = (A ИЛИ B) И C или Y = A ИЛИ (B И C)? **В булевых уравнениях наибольший приоритет имеет операция НЕ, затем идет И, затем ИЛИ.** Как и в обычных уравнениях, произведения вычисляются до вычисления сумм. Таким образом, правильно уравнение читается как Y = A ИЛИ (B И C).

ДИЗЪЮНКТИВНАЯ ФОРМА

Таблица истинности для функции N переменных содержит 2^N строк, по одной для каждой возможной комбинации значений входов. Каждой строке в таблице истинности соответствует минтерм, который имеет значение ИСТИНА для этой строки. На Рис. ниже показана таблица истинности функции двух переменных A и B. **В каждой строке показан соответствующий ей минтерм**. Например, минтерм для первой строки – это $\overline{A}$ $\overline{B}$, поскольку $\overline{A}$ $\overline{B}$ имеет значение ИСТИНА тогда, когда A = 0 и B = 0. Минтермы нумеруют начиная с 0; первая строка соответствует минтерму 0 (m0), следующая строка – минтерму 1 (m1), и так далее.

A	B	Y	minterm	minterm name
0	0	0	$\overline{A}$ $\overline{B}$	m_0
0	1	1	$\overline{A}$ B	m_1
1	0	0	A $\overline{B}$	m_2
1	1	0	A B	m_3

Рис. Таблица истинности и минтермы

Можно написать **булево уравнение для любой таблицы истинности** путем **суммирования всех тех минтермов, для которых выход Y имеет значение ИСТИНА**. Например, на Рис. выше есть только

одна строка (минтерм), для которой выход Y имеет значение ИСТИНА, она отмечена овалом. Таким образом, Y = $\overline{A}$B.

На Рис. ниже показана таблица, в которой выход имеет значение ИСТИНА для нескольких строк. Суммирование отмеченных минтермов дает Y = $\overline{A}$B+AB. **Такая сумма минтермов называется совершенной дизъюнктивной нормальной формой функции (sum-of-products canonical form). Далее по тексту я могу вместо фразы писать сокращение СДНФ.**

СДНФ - представляет собой **сумму (операцию «ИЛИ») произведений (операций «И», образующих минтермы).**

Совершенная дизъюнктивная нормальная форма СДНФ также может быть записана через символ суммы Σ. При использовании такого обозначения функция на Рис. ниже будет выглядеть так:

$F(A, B) = \Sigma(m_1, m_3)$

или

$F(A, B) = \Sigma(1, 3)$

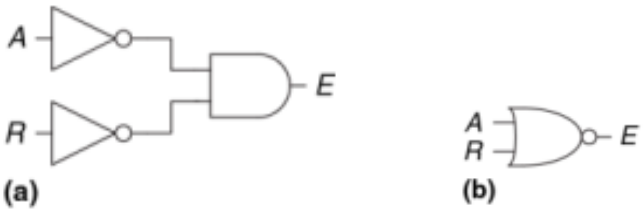
A	B	Y	minterm	minterm name
0	0	0	$\overline{A}\,\overline{B}$	m_0
0	1	1	$\overline{A}\,B$	m_1
1	0	0	$A\,\overline{B}$	m_2
1	1	1	$A\,B$	m_3

Рис. Таблица истинности с несколькими минтермами, равными ИСТИНЕ

Пример: A – вы не сделали вовремя и не защити ЛР, R – вы пропускали лекции, E – наличие зачета по ВМСИС. Таблица истинности и схема представлены ниже.

A	R	E
0	0	1
0	1	0
1	0	0
1	1	0

Булево уравнение или СДНФ для данного примера: E = $\overline{A}\,\overline{R}$. Реализация представлена на рисунке ниже под буквой a). Но схема под буквой b) даст такой же результат. Схема под буквой b) описывается выражением: E = $\overline{A + R}$. Согласны?



Совершенная дизъюнктивная нормальная форма СДНФ позволяет записать булево уравнение для любой таблицы истинности с любым количеством переменных. Но не всегда позволяет получить простое уравнение.

КОНЪЮНКТИВНАЯ ФОРМА

Альтернативный способ выражения булевых функций – это **совершенная конъюнктивная нормальная форма (products-of-sum forms) - СКНФ. Каждая строка таблицы истинности соответствует макстерму, который имеет значение ЛОЖЬ для этой строки.**

Например, макстерм для первой строки для двухвходовой таблицы истинности – это (A + B), поскольку (A + B) имеет значение ЛОЖЬ, когда A = 0 и B = 0. Для любой схемы, заданной таблицей истинности, мы можем записать ее **булево уравнение как логическое «И» всех макстермов, для которых выход имеет значение ЛОЖЬ.** Совершенная конъюнктивная нормальная форма также может быть записана с использованием символа П.

A	B	Y	maxterm	maxterm name
0	0	0	A + B	M_0
0	1	1	A + $\overline{B}$	M_1
1	0	0	$\overline{A}$ + B	M_2
1	1	1	$\overline{A}$ + $\overline{B}$	M_3

Таблица истинности имеет две строки, в которых выход имеет значение ЛОЖЬ. Следовательно, функция может быть записана в конъюнктивной форме так: Y = (A + B) ($\overline{A}$ + B). Также функция может быть записана как Y = П(M0, M2) или Y = П(0, 2). Первый макстерм, (A + B), гарантирует, что Y = 0 для A = 0 и B = 0, так как логическое «И» любого значения и нуля дает нуль. Аналогично, второй макстерм ($\overline{A}$ + B) гарантирует, что Y = 0 для комбинации A = 1 и B = 0.

Эта таблица по исходным данным идентична той, что смотрели в разделе ранее «ДИЗЪЮНКТИВНАЯ ФОРМА», но булево выражение в итоге записано по-разному, а значит и реализация на схеме будет разная.

В примере с зачетом по ВМСИС также можно итоговое уравнение записать через СКНФ иначе:
E = (A + $\overline{R}$)($\overline{A}$ + R)($\overline{A}$ + $\overline{R}$) или E = П(1, 2, 3).

A	R	E
0	0	1
0	1	0
1	0	0
1	1	0

Эти уравнения логически **эквивалентны.**

Дизъюнктивная форма дает более **короткое уравнение**, когда выход имеет значение ИСТИНА только в нескольких строках таблицы истинности; конъюнктивная же форма проще, когда выход имеет значение ЛОЖЬ только в нескольких строках таблицы истинности.

БУЛЕВЫ УРАВНЕНИЯ ИДЕАЛЬНО ПОДХОДЯТ ДЛЯ ОПИСАНИЯ ЦИФРОВОЙ ЛОГИКИ.

И главной задачей является МИНИМИЗАЦИЯ ЛОГИЧЕСКИХ ФУНКЦИЙ.

Сложность логической функции, а отсюда сложность и стоимость реализующей ее схемы (цепи), пропорциональны числу логических операций и числу вхождений переменных или их отрицаний. В принципе любая логическая функция может быть упрощена непосредственно с помощью аксиом и теорем логики, но, как правило, такие преобразования требуют громоздких выкладок.

К тому же процесс упрощения булевых выражений не является алгоритмическим. Поэтому более целесообразно использовать специальные алгоритмические методы минимизации, позволяющие проводить упрощение функции более просто, быстро и безошибочно.

К таким методам относятся: **метод Квайна, метод карт Карно, метод испытания импликант, метод импликантных матриц, метод Квайна-Мак-Класки и др.**

Эти методы наиболее пригодны для обычной практики, особенно минимизация логической функции с использованием карт Карно. **Метод карт Карно сохраняет наглядность при числе переменных не**

более шести. В тех случаях, когда число аргументов больше шести, обычно используют метод Квайна-Мак-Класки.

МИНИМИЗАЦИЯ ЛОГИЧЕСКИХ ФУНКЦИЙ ПРИ ПОМОЩИ КАРТ КАРНО

Карта Карно — графический способ минимизации переключательных (булевых) функций.

Карты Карно были изобретены в 1952 Эдвардом В. Вейчем и усовершенствованы в 1953 Морисом Карно, физиком из «Bell Labs», и были призваны помочь упростить цифровые электронные схемы. Да, это углубленно проходят схемотехники.

В карту Карно булевы переменные передаются из таблицы истинности и упорядочиваются с помощью кода Грея, в котором каждое следующее число отличается от предыдущего только одним разрядом.

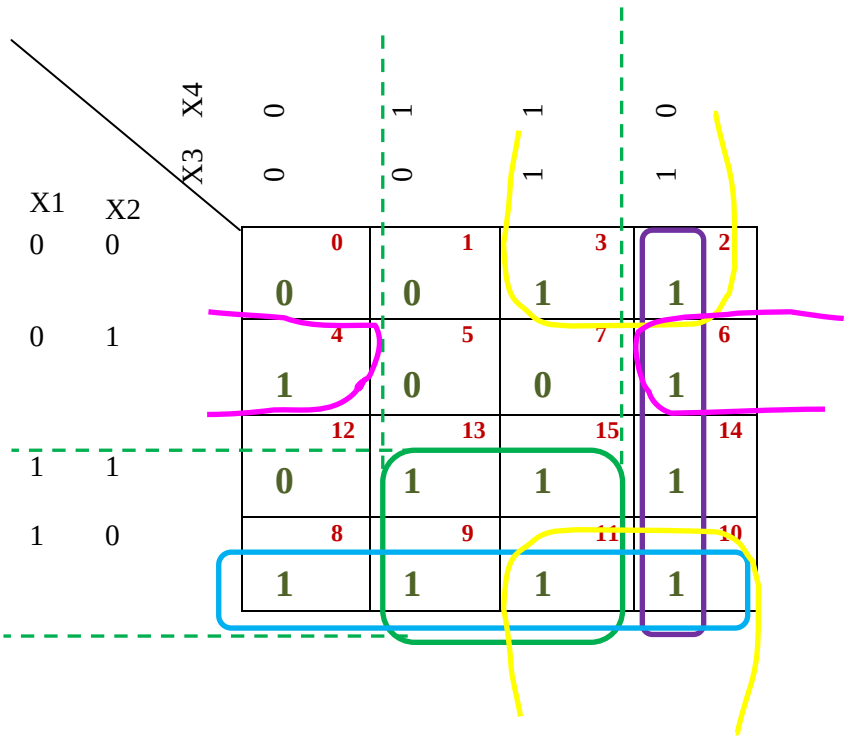
Пример №1:

Как на карту Карно можно нанести булевую функцию? Таблица двоичных чисел и карта Карно от 4 булевых переменных.
У нас 4 входных параметра, поэтому 2 в степени 4 = 16 неповторяющихся комбинаций, при которых, например выход будет как в последнем столбце:

№	X1	X2	X3	X4	F (X1, X2,X3,X4)
0	0	0	0	0	0
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

Карты Карно могут быть от 2,3,4,5 и возможно даже сделать от 6 переменных. При большем количестве переменных метод не удобен.
На карте есть координаты от X1 до X4 (00, 01, 11, 10) они всегда такие у любой карты от 4 переменных Карно выглядят именно так и в такой последовательности *(чуть позже по тексту станет яснее)*. У каждой клетки как в шахматах есть свой порядковый номер (отмечен красным на примере ниже). Нумерация клеток идет не по порядку с помощью десятичных чисел, что связано с тем, что **координаты клеток в двоичной системе (код Грея) также идет не по порядку ($00_2=0_{10}$, $01_2=1_{10}$, $11_2=3_{10}$, $10_2=2_{10}$). Координаты соседние друг с другом должны отличаться друг от друга только на 1 разряд (00 отличается от 01 на 1 во втором разряде, 01 от 11 на 1 в первом разряде и т.д.), т.е. каждое последующее двоичное число отличается от предыдущего только в одном разряде (это правило идет из описания граней n-мерного куба)** *(чуть позже по тексту станет яснее).*

Значение булевой функции нанесено в карте Карно ниже зеленым цветом.



Теперь нужно из карты перейти к белевому выражению, причем в сокращенной (минимизированной форме). Для этого нужно поработать с контурами.

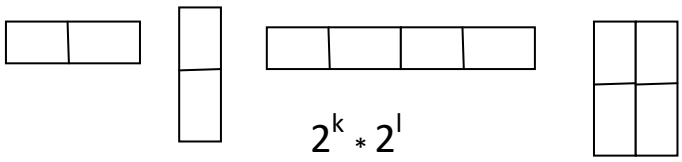
Переменные выделяются в контуры (работаем либо с 0, либо с 1).

Есть КНФ (конъюктивно нормальная форма), конъюнкция – логическое умножение (И). Работаем с 0 на карте Карно.

Есть ДНФ (дизъюнктивно нормальная форма). Дизъюнкция – логическое сложение (ИЛИ). Работаем с 1 на карте Карно.

Давайте сейчас возьмем формат ДНФ **(какой формат взять вам – ваше личное решение и на работу функции это влияния не окажет).** Объединяем в контуры единицы.

- Контуры могут быть квадратами или прямоугольниками. Содержать 1,2,4 или 8 клеток (2^0, 2^1, …).



- Контуры могут пересекаться.
- Чем больше контур, тем лучше.
- Лишние контура брать НЕ надо, теряется смысл минимизации итоговой функции (совершенной функции).
- **Карта Карно – это развертка n-мерного куба, хотя некоторые говорят, что это развертка на плоскость сферы, тора, т.е. можно брать контура через край.**

У выбранного контура есть координаты, для зеленого контура проведены пунктирные линии к его координатам.

Мы выбрали ДНФ, что повлияло на поиск контуров внутри карты по единицам, так и повлияет на описание контура (зеленого, голубого, розового и т.д.) с помощью его координат (X1,X2,X3,X4).

Например, **розовый контур** описывается как: $\overline{X1} \cdot X2 \cdot \overline{X4}$.
Почему именно так описывается контур? При описании контура мы пишем его координаты. Берем координату в явном виде, если она не изменялась на протяжении всего контура и была равна 1 в случае данного рассматриваемого примера (мы взяли ДНФ), как координата X2. Мы берем «НЕ» координату (черта сверху), если координата не изменялась при описании контура, но была равна 0. Мы не пишем координату вовсе, если она менялась на протяжении описания контура с 0 на 1 или наоборот, как в случае X3.
Фиолетовый контур: координаты X1, X2 изменяются, значит не берем при описании контура. X3 и X4 не изменяются, поэтому берем при описании контура, но тк описываем по ДНФ, то $\overline{X4} \cdot X3$.
Голубой контур: $\overline{X2} \cdot X1$
Желтый контур: $\overline{X2} \cdot X3$.
Зеленый контур: $X1 \cdot X4$

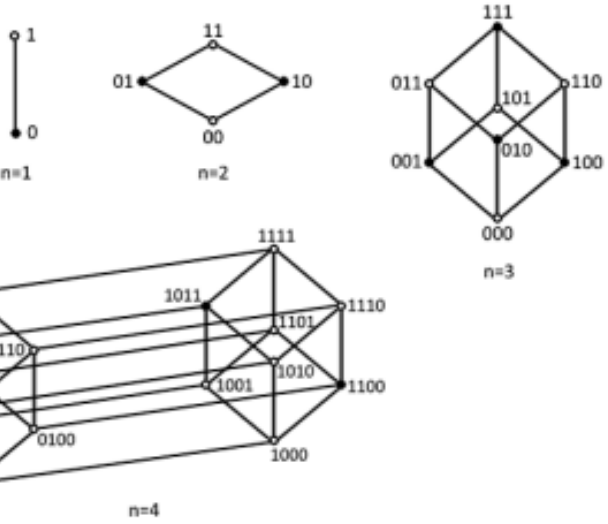
Можно себя проверить, правильным ли количеством координат (иксов) вы описали контур:
Контур из одной клетки описывается четырьмя иксами, контур из двух клеток – тремя иксами, контур из четырех клеток – двумя иксами, из восьми клеток – одним иксом.

Итак, результат минимизации булевой функции (функция будет результатом сложения (ИЛИ) функций, описывающих наши контуры, т.к. мы выбрали Дизъюнкцию): $F(X1, X2, X3, X4)_{СДНФ} = \overline{X1} \cdot X2 \cdot \overline{X4} + \overline{X4} \cdot X3 + \overline{X2} \cdot X1 + \overline{X2} \cdot X3 + X1 \cdot X4$,
Где СДНФ – совершенная дизъюнктивная нормальная форма

В примере №1 функцию F упрощали с помощью карты Карно. Но более наглядным (для некоторых людей) является минимизация булевых функций геометрическим методом на n-мерном единичном кубе. Примеры единичных кубов и соответствующие им функции на скрине ниже:

Примеры булевых функций для построения -мерных единичных кубов.

$n = 1$	$\overline{x}$
$n = 2$	$x + y$
$n = 3$	$x + y + z$
$n = 4$	$\overline{x_1}x_2\overline{x_3}x_4 \vee x_1\overline{x_2}x_3x_4 \vee x_1x_2\overline{x_3x_4}$

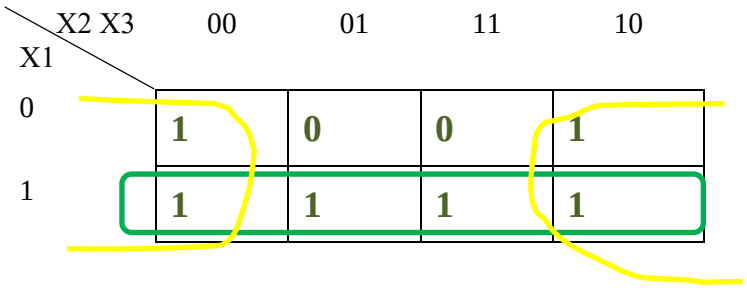


Но очевидно, что даже 4-мерный куб (от 4-х переменных) изобразить очень сложно, не говоря уже о большем количестве переменных. Поэтому второй пример (хоть в нем всего 3 входных переменных) мы также минимизируем с помощью карты Карно.

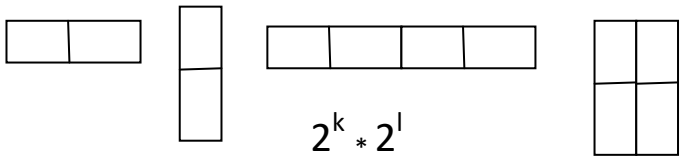
Пример №2:

У нас 3 входных параметра, поэтому 2 в степени 3 = 8 неповторяющихся комбинаций, при которых, например, выход будет как в последнем столбце.

№	X1	X2	X3	F (X1, X2,X3)
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



Ну что ж, покроем нашу карту Карно прямоугольниками, стороны которых имеют некоторую степень двойки:



Если мы снова, как в примере №1, решим минимизировать функцию F по ДНФ (дизъюнктивно нормальная форма). Дизъюнкция – логическое сложение (ИЛИ). Работаем с 1 (единицами) на карте Карно.

То СДНФ (совершенная дизъюнктивная нормальная форма) функции F (X1, X2,X3) $_{\text{СДНФ}} = \overline{X3} + X1$ или запись $X1 \; V \; \overline{X3}$.

$V -$ *знак* $+$, *логическое ИЛИ;*

$\Lambda -$ *знак* $*$, *логические И*

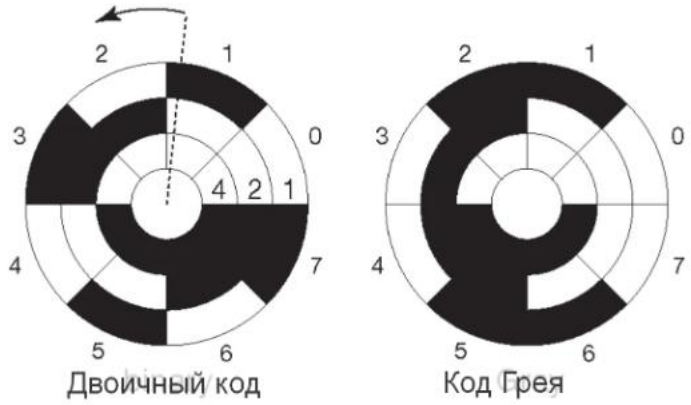
n-мерные единичные кубы и карты Карно удобны при минимизации функций вручную, в случае если задача стоит запрограммировать функцию или количество входных переменных больше 6, то оба способа не удобны и применяют алгоритм Куайна—Мак-Класки (Квайна –МакКласки).

Ранее в тексте встречалось упоминание **«Код Грея».** Код назван в честь Фрэнка Грея, который в 1947 году получил патент на «отражённый двоичный код».
Двоичный код может быть оптимизирован под нужные нам условия. Для этого как раз и существует код Грея.

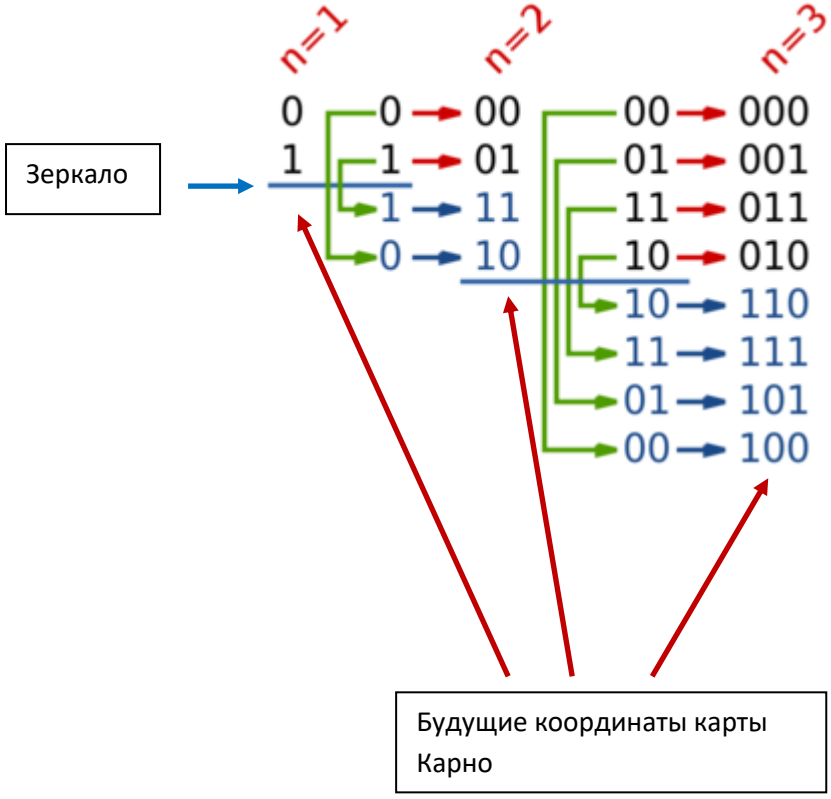
N	Двоичный код	Код Грея
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Это тот же двоичный код, но порядок чисел в нём идёт особым образом. Основная идея в том, что следующее число в коде получается из предыдущего сменой только одного символа. Этот код является помехозащищённым. Обратите внимание на любую строчку кода Грея, любые две строчки отличаются друг от друга не более чем на один символ. В обычном двоичном коде при переход от 3 к 4 изменяются все символы 011->100. В коде Грея это будет 010 -> 110.
За счет этой особенности мы понимаем, что если изменилось 2 или более символов, то последовательность не совпадёт, а значит мы обнаружим ошибку.

Код Грея, например, широко используется в угловых энкодерах.



Вы скажете, что код Хемминга гораздо более помехозащищенный. Он позволяет не только обнаружить, но и восстановить единичную ошибку. Применение кодов Хемминга в цифровых преобразователях угла на основе псевдослучайных кодовых шкал также присутствует.



Спектр применения кодов Грея обширен, возможно, для Вас будет ближе объяснение, связанное с пройденным предметом «Электроника»: при проектировании проектов на **ПЛИС (Программируемая логическая интегральная схема),** например, если создать большой счетчик на 20 триггерах, то при его инкрементировании обязательно возникают ситуации, когда сразу несколько триггеров меняют свое состояние. При этом триггеры переключаются в разное время и это очень плохо, стабильность такого счетчика оставляет желать лучшего и на больших частотах схема может работать неправильно. Если же использовать коды Грея, то такой проблемы не будет, т.к. в соседних состояниях отличным может быть только один разряд. Т.е. только один триггер переключится, что исключает так называемую «гонку сигналов».

- ✓ Выдать индивидуальное задание.
- ✓ Рассказать о логическом автомате
- ✓ Рассказать о теоремах и аксиомах
- ✓ Сказать применить на своем задании теорему де Моргана
- ✓ Рассказать о сборке в программе
- ✓ Триггеры
- ✓ Транзисторы?

Тестовое для пары

№	x1	x2	x3	x4	F (x1, x2, x3, x4)
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	1
11	1	0	1	1	1
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1