Министерство науки и высшего образования Российской Федерации

Томский государственный университет
систем управления и радиоэлектроники

Д. В. Озеркин

ПРОГРАММНЫЙ КОМПЛЕКС СХЕМОТЕХНИЧЕСКОГО МОДЕЛИРОВАНИЯ MICROCAP

Методические указания для лабораторных работ
по дисциплинам «Информатика»,
«Информационные технологии в электронике»
для направления подготовки
110303 Конструирование и технология электронных средств

Томск
2022

УДК 004.91
ББК 32.85
О-46

Озеркин, Денис Витальевич
Программный комплекс схемотехнического моделирования MicroCAP : Методические указания для лабораторных работ по дисциплинам «Информатика», «Информационные технологии в электронике» для направления подготовки 110303 Конструирование и технология электронных средств / Д. В. Озеркин. – Томск: Томск. гос. ун-т систем упр. и радиоэлектроники, 2022. – 105 с.

В настоящих методических указаниях по дисциплинам «Информатика» и «Информационные технологии в электронике» основным «инструментом» для выполнения лабораторных работ выступает программа схемотехнического моделирования MicroCAP.

Основное назначение методических указаний – изучение принципов построения и проектирования функциональных узлов и устройств ЭВМ, а также цифровой автоматики.

Методические указания предназначены для студентов, обучающихся по направлению 110303 Конструирование и технология электронных средств.

Одобрено на заседании кафедры РЭТЭМ протокол № 78 от 16.02.2022.

УДК 004.91
ББК 32.85

ОГЛАВЛЕНИЕ

НАЗНАЧЕНИЕ МЕТОДИЧЕСКИХ УКАЗАНИЙ

К настоящему времени наиболее совершенные принципы и средства взаимодействия человека с окружающим миром обеспечила цифровая техника. Ее наименее избыточный алфавит – двухуровневые символы, которыми оказалось возможным представить (кодировать) любую информацию – привел к созданию чрезвычайно точных, надежных, малогабаритных и функционально-наращиваемых устройств.

Использование в цифровой технике двухсимвольного алфавита привело к созданию новых, исключительно эффективных методов передачи, хранения и преобразования сигналов, к новым средствам обработки информации – компьютерным технологиям. Под этим словосочетанием понимают технологию обработки с использованием современных средств цифровой техники и ее вершины – вычислительной техники. Так родились основанные на новых принципах современные технологии: связи (цифровая связь и цифровое телевидение), обнаружения (цифровая радиолокация и цифровая навигация), вычислений и автоматического управления (электронно-вычислительная техника) и т.д.

Цифровая техника стоит на трех «китах». Первый «кит» - теорема о дискретизации, сформулированная и доказанная в 1933 г. академиком В.А. Котельниковым. В этой теореме теоретически обоснована возможность получения цифрового эквивалента (цифрового образа) аналогового сигнала, хранить, передавать и обрабатывать который оказалось значительно проще и точнее, чем осуществлять аналогичные действия над аналоговым сигналом.

Второй «кит» – алгебра логики (булева алгебра, названная так в честь ее автора – ирландского математика Дж. Буля). Получившая дальнейшее развитие в основополагающих трудах А.Н. Колмогорова, К. Шеннона, В.И. Шестакова и других ученых, алгебра логики позволила поставить анализ и синтез цифровых схем на прочный математический фундамент.

Третий «кит» - импульсная техника, из которой цифровая техника заимствовала многие принципы, элементы и устройства.

Цифровые устройства обладают рядом преимуществ перед аналоговыми: огромная степень интеграции, составляющая десятки миллионов транзисторов в одной микросхеме, чрезвычайно низкая погрешность, малая зависимость от параметров окружающей среды.

Области применения цифровой техники поистине безграничны. К сказанному ранее можно добавить, что в настоящее время до 90% всех разрабатываемых устройств – цифровые. Со знанием цифровой техники будущий инженер окажется востребованным в любой области и сумеет внести вклад в ее развитие.

При изучении электротехнических дисциплин в последние годы широко применяются программы и системы, позволяющие проводить схемотехническое моделирование. Внедрение в учебный процесс компьютерного моделирования обусловлено рядом причин. Среди них можно выделить такие, как:

- экономическая целесообразность. Модернизация устаревшего оборудования лабораторий в ряде случаев дороже оборудования компьютерного класса;
- возможность исследования схем, содержащих самые современные компоненты электронных схем;
- возможность проведения исследования схем и их компьютерного моделирования в домашних условиях;
- возможность дистанционного обучения;
- программные продукты, используемые при моделировании электронных схем, как правило, не требуют специальных настроек и нестандартного оборудования. Для установки таких программ и систем могут быть использованы компьютеры простых конфигураций.

В настоящих методических указаниях для целей моделирования используется программный комплекс MicroCAP. Выбор указанного программного продукта не случаен, а продиктован следующими основными причинами:

- многолетний опыт применения различных версий программного продукта в учебном процессе;
- минимальные аппаратные требования к персональному компьютеру, о чем уже говорилось выше;
- исключительно схемотехническая направленность программного продукта, сбалансированность для целей настоящего лабораторного практикума. Следует отметить, что другие программные комплексы – OrCAD, MultiSim (EWB), Altium Designer – наряду со схемотехническим моделированием обладают возможностями топологического проектирования. В данном случае этот факт можно считать недостатком, поскольку указанные программные комплексы гораздо сложнее в освоении студентами.

Основное назначение методических указаний – изучение принципов построения и проектирования функциональных узлов и устройств ЭВМ, а также цифровой автоматики. Кроме этого, методические указания способствуют приобретению студентами, обучающимися по направлению подготовки 110303 «Конструирование и технология электронных средств» серьезных практических навыков в схемотехническом моделировании цифровых устройств.

ЛАБОРАТОРНАЯ РАБОТА №1 – СИНТЕЗ КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ

1.1 Цель работы

В ходе выполнения настоящей работы предусматривается:

1) знакомство с основными этапами синтеза комбинационных логических устройств;

2) изучение способов минимизации логических функций, представленных в совершенной дизъюнктивной форме;

3) приобретение навыков перехода в логический базис элементов И-НЕ, ИЛИ-НЕ;

4) построение структурных схем логических устройств по заданным функциям алгебры логики.

1.2 Порядок выполнения работы

1. Изучить методические указания к лабораторной работе.
2. Письменно, в отчете по лабораторной работе ответить на контрольные вопросы.
3. Внимательно ознакомиться с методическим примером, приведенным в пункте 1.4.
4. Выполнить лабораторное задание согласно варианту задания.
5. Сделать выводы по работе.

Внимание! Отчет по лабораторной работе в обязательном порядке должен содержать: схемы включения, графики зависимостей, все необходимые расчеты и их результаты, текстовые пояснения. На графиках в отчете должны присутствовать единицы измерения, масштаб, цена деления.

1.3 Канонические формы представления логических функций. Минимизация логических функций. Синтез логических устройств в заданных базисах

Синтез логического устройства распадается на несколько этапов. На первом этапе функцию, заданную в словесной, табличной или другой формах, требуется представить в виде логического выражения с использованием некоторого базиса. Дальнейшие этапы сводятся к получению минимальных форм функций, обеспечивающих при синтезе наименьшее количество электронного оборудования и рациональное построение функциональной схемы устройства. Для первого этапа обычно используется базис И, ИЛИ, НЕ независимо от базиса, который будет использован для построения логического устройства.

Для удобства последующих преобразований приняты следующие две исходные канонические формы представления функций: совершенная дизъюнктивная нормальная форма (СДНФ) и совершенная конъюнктивная нормальная форма (СКНФ).

Дизъюнктивной нормальной формой (ДНФ) называется такая форма представления функции, при которой логическое выражение функции строится в виде дизъюнкции ряда членов, каждый из которых является простой конъюнкцией аргументов или их инверсией. Примером ДНФ может служить выражение:

$$f(x_3, x_2, x_1) = x_1 + \bar{x}_3 \cdot x_2 + x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + x_3 \cdot x_2 . \quad (1.1)$$

Приведем форму представления функции, не являющуюся ДНФ. Например, функция

$$f(x_3, x_2, x_1) = x_1 + \bar{x}_3 \cdot x_2 + x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + \overline{x_3 \cdot x_2}$$

представлена не в ДНФ, так как последний член не является простой конъюнкцией аргументов. Также не является ДНФ следующая форма представления функции:

$$f(x_3, x_2, x_1) = x_1 \cdot (\bar{x}_3 \cdot x_2 + x_3 \cdot \bar{x}_2) + x_3 \cdot x_2.$$

Если в каждом члене ДНФ представлены все аргументы (или их инверсии) функции, то такая форма называется СДНФ. Выражение (1.1) не является СДНФ, так как в нем лишь третий член содержит все аргументы функции.

Для перехода от ДНФ к СДНФ необходимо в каждый из членов, в которых представлены не все аргументы, ввести выражение вида $x_i + \bar{x}_i$, где x_i – отсутствующий в члене аргумент. Так как $x_i + \bar{x}_i = 1$, такая операция не может изменить значений функции. Покажем переход от ДНФ к СДНФ на примере простого выражения:

$$f(x_3, x_2, x_1) = x_1 + \bar{x}_3 \cdot x_2.$$

Добавление в члены выражений вида $x_i + \bar{x}_i$ приведет к функции:

$$f(x_3, x_2, x_1) = x_1 \cdot (x_2 + \bar{x}_2) \cdot (x_3 + \bar{x}_3) + \bar{x}_3 \cdot x_2 \cdot (x_1 + \bar{x}_1) =$$

$$= x_3 \cdot x_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot \bar{x}_1.$$

На основании тождества алгебры логики $x + x = x$ имеем:

$$\bar{x}_3 \cdot x_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot x_1 = \bar{x}_3 \cdot x_2 \cdot x_1.$$

Отсюда после приведения подобных членов:

$$f(x_3, x_2, x_1) = x_3 \cdot x_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot \bar{x}_1,$$

т.е. имеем СДНФ.

Если исходная функция задана в табличной форме, то СДНФ может быть получена непосредственно. Пусть задана функция в форме таблицы 1.1.

Для этой функции СДНФ имеет вид:

$$f(x_3, x_2, x_1) = \bar{x}_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_2 \cdot x_1 + x_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot x_2 \cdot x_1. \qquad (1.2)$$

Каждый член в (1.2) соответствует некоторому набору значений аргументов, при котором $f(x_3, x_2, x_1)$ равна *1*. Каждый из наборов аргументов, при которых $f(x_3, x_2, x_1)$ равна *1* (третий, четвертый, шестой и восьмой столбцы наборов), обращает в единицу соответствующий член выражения (1.2), вследствие чего и вся функция оказывается равной единице.

Таблица 1.1 – Таблица истинности логической функции трех аргументов

X_3	X_2	X_1	$F(X_3, X_2, X_1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Можно сформулировать следующее правило записи СДНФ функции, заданной таблицей истинности. Необходимо записать столько членов в виде конъюнкций всех аргументов, сколько единиц содержит функция в таблице. Каждая конъюнкция должна соответствовать определенному набору значений аргументов, обращающему функцию в единицу, и если в этом наборе значение аргумента равно нулю, то в конъюнкцию входит инверсия данного аргумента. Следует отметить, что любая функция имеет единственную СДНФ.

Структурная схема логического устройства может быть построена непосредственно по канонической форме (например, СДНФ) реализуемой функции. Получающаяся при этом

схема для функции (1.2) показана на рисунке 1.1. Недостаток такого метода построения структурных схем, обеспечивающего в общем правильное функционирование устройства, состоит в том, что получающиеся схемы чаще всего неоправданно сложные, требуют использования большого числа логических элементов, имеют низкие экономичность и надежность. Во многих случаях удается так упростить логическое выражение, не изменив функции, что соответствующая структурная схема оказывается существенно более простой. Методы такого упрощения функции называются методами *минимизации функций*.

Наиболее просто и наглядно *задача минимизации логической функции* решается с использованием ее *кубического представления*. Такое представление использует ограниченное число символов и поэтому применяется при автоматизации процессов логического проектирования цифровых интегральных схем.

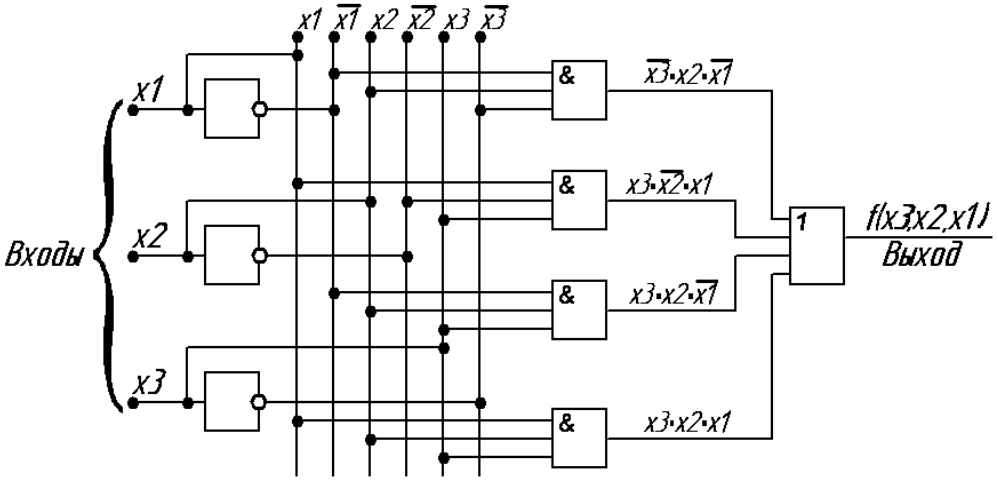


Рисунок 1.1 – Структурная схема устройства, построенная по СДНФ

Основой кубической формы является представление каждого набора входных переменных в качестве *n*-мерного вектора. Вершины этих векторов геометрически могут быть представлены как вершины *n*-мерного куба. Отмечая точками вершины векторов, для которых логическая функция равна единице, получаем геометрическое представление функции в виде куба. Пусть дана логическая функция трех аргументов:

$$f(x_3, x_2, x_1) = x_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot x_2 \cdot x_1.$$

Графический образ данной функции в виде куба представлен на рисунке 1.2, *а*.

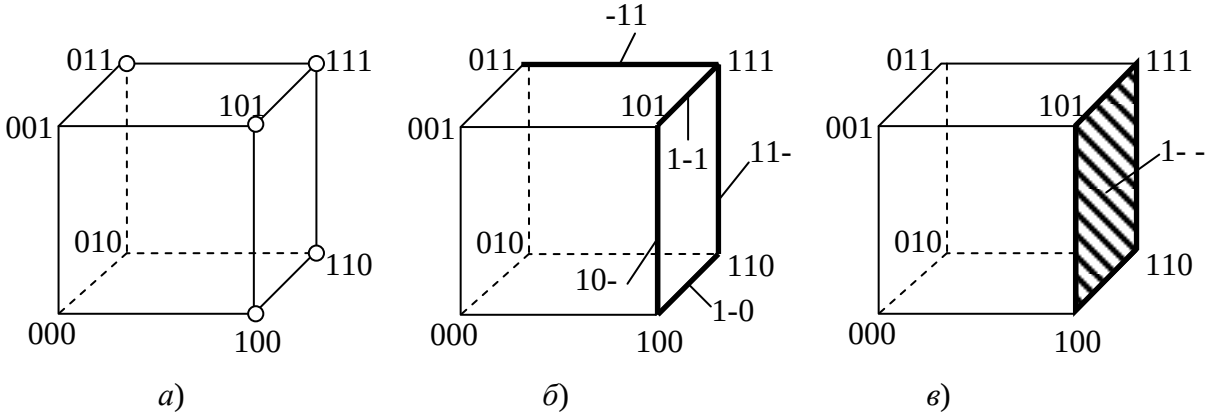


Рисунок 1.2 – Геометрическое представление логической функции:
а) нулевой кубический комплекс; *б*) единичный кубический комплекс;
в) двоичный кубический комплекс

Очевидно, что наборы переменных, расположенные на концах ребер куба, отличаются только одной переменной. Такие наборы (коды) принято называть *соседними*. Каждую вершину куба, в которой функция принимает единичное значение, называют *нулевым кубом* (0-кубом). Записывается 0-куб последовательностью образовавших его входных переменных, т.е. кодом, соответствующим единице в таблице истинности. Множество нулевых кубов образуют нулевой кубический комплекс. Например, K_0 = (011, 100, 101, 110, 111) (рисунок 1.2, *а*).

Если два нулевых куба комплекса K_0 отличаются только по одной координате (переменной), т.е. два набора переменных, для которых логическая функция равна единице, являются соседними, то они образуют единичный куб (1-куб). Геометрически это соответствует ребру исходного *n*-мерного куба (рисунок 1.2, *б*), 1-куб записывается последовательностью общих элементов образовавших его 0-кубов с прочерком несовпадающих элементов. Множество единичных кубов образуют единичный кубический комплекс. Например, K_1 = (-11, 10-, 1-0, 11-, 1-1).

Аналогично, если два единичных комплекса K_1 отличаются только по одной координате (переменной), то эти единичные кубы образуют двоичный куб (2-куб). Геометрически это соответствует грани исходного *n*-мерного куба (рисунок 1.2, *в*). 2-куб также записывается последовательностью общих элементов, образовавших его 1-кубов с прочерком несовпадающих элементов, а множество двоичных кубов образуют двоичный кубический комплекс. Например, K_2 = (1- -). И так далее.

Из сказанного следует, что размерность куба (его *ранг*) определяется числом несовпадающих координат, т.е. числом прочерков в его записи. Объединение кубических комплексов K_0, K_1, …, K_m для логической функции *n* переменных образует ее кубический комплекс:

$$K(f) = \cup (K_0, K_1, \ldots, K_m).$$

Например, кубический комплекс логической функции, представленный на рисунке 1.2, имеет вид:

$$K(f) = (011, 100, 101, 110, 111, -11, 10-, 1-0, 11-, 1-1, 1- -).$$

Из кубического комплекса K(*f*) всегда можно выделить множество кубов П(*f*), таких, что каждый член комплекса K_0, т.е. вершина куба будет включен по крайней мере в один куб из множества П(*f*). Множество кубов П(*f*) называется *покрытием* комплекса K(*f*) или покрытием логической функции. Очевидно, что для любой логической функции существует несколько ее покрытий. В свою очередь, каждому покрытию П(*f*), так же как и самому комплексу, соответствует своя дизъюнктивная нормальная форма, получаемая логическим суммированием логических произведений, соответствующих выделенным кубам функции.

Например, нулевой кубический комплекс (рисунок 1.2, *а*) включает все вершины куба, поэтому образует покрытие функции:

$$П_1(z) = K_0 = (011, 100, 101, 110, 111).$$

Все вершины куба включаются также в единичный кубический комплекс K_1, поэтому он также образует покрытие функции:

$$П_2(z) = K_1 = (-11, 10-, 1-0, 11-, 1-1).$$

Перебирая сочетания кубов различных рангов можно получить следующие покрытия функции:

$$П_3(z) = (011, 11-, 10-);$$
$$П_4(z) = (-11, 1-1, 1-0);$$
$$П_5(z) = (011, 1- -);$$
$$П_6(z) = (-11, 1- -) \text{ и так далее.}$$

Соответствующие указанным покрытиям ДНФ имеют вид:

$$f_1(x) = \bar{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_2 \cdot x_1 + x_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot x_2 \cdot x_1;$$

$$f_2(x) = x_2 \cdot x_1 + x_3 \cdot \bar{x}_2 + x_3 \cdot \bar{x}_1 + x_3 \cdot x_2 + x_3 \cdot x_1;$$

$$f_3(x) = \bar{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot x_2 + x_3 \cdot \bar{x}_2;$$
$$f_4(x) = x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot \bar{x}_1;$$
$$f_5(x) = \bar{x}_3 \cdot x_2 \cdot x_1 + x_3;$$
$$f_6(x) = x_2 \cdot x_1 + x_3.$$

Сложность полученной таким образом ДНФ принято характеризовать понятием «цена покрытия» ($Ц_П$), которая равна сумме цен всех кубов, составляющих данное покрытие П(*f*): $Ц_П = \Sigma\, Ц_К$. В свою очередь, цена одного *r*-куба функции *n*-переменных определяется как разность полного числа входных переменных и ранга соответствующего куба, т.е. равна числу переменных в соответствующей дизъюнкции: $Ц_К = n - r$. Так, для функции трех переменных цена 0-куба равна трем, а 2-куба – единице.

В соответствие со сказанным, задача минимизации логической функции сводится к поиску покрытия П(*f*) кубического комплекса К(*f*), имеющего минимальную цену.

Например, среди перечисленных выше покрытий выражение $П_6(z) =$ (-11, 1- -) имеет минимальную цену:

$$Ц_{К1} = 3 - 1 = 2;\ Ц_{К2} = 3 - 2 = 1;\ Ц_П = \Sigma\, Ц_К = 2 + 1 = 3.$$

Следовательно, логическая функция $f_6(x) = x_2 \cdot x_1 + x_3$ является минимальной дизъюнктивной нормальной формой (МДНФ).

Метод минимизации функции с помощью карт Вейча обеспечивает простоту получения результатов. Он используется при минимизации относительно несложных функций (с числом аргументов до пяти) ручным способом. Этот метод требует изобретательности и не может быть применен для решения задачи минимизации с помощью ЭВМ. Карта Вейча представляет собой определенную форму таблицы истинности. Таблицы, представленные на рисунке 1.3, являются картами Вейча для функций двух, трех и четырех аргументов.

Каждая клетка карты соответствует некоторому набору значений аргументов. Этот набор аргументов определяется присвоением значения *логическая 1* буквам, на пересечении строк и столбцов которых расположена клетка. Так, в карте функций четырех аргументов (рисунок 1.3, *в*) клетки первой строки соответствуют следующим комбинациям значений аргументов:

- первая клетка $x_1 = 1$, $x_2 = 1$, $\bar{x}_3 = 1$ ($x_3 = 0$), $\bar{x}_4 = 1$ ($x_4 = 0$);
- вторая клетка $x_1 = 1$, $x_2 = 1$, $x_3 = 1$, $x_4 = 0$;
- третья клетка $x_1 = 1$, $x_2 = 0$, $x_3 = 1$, $x_4 = 0$;
- четвертая клетка $x_1 = 1$, $x_2 = 0$, $x_3 = 0$, $x_4 = 0$.

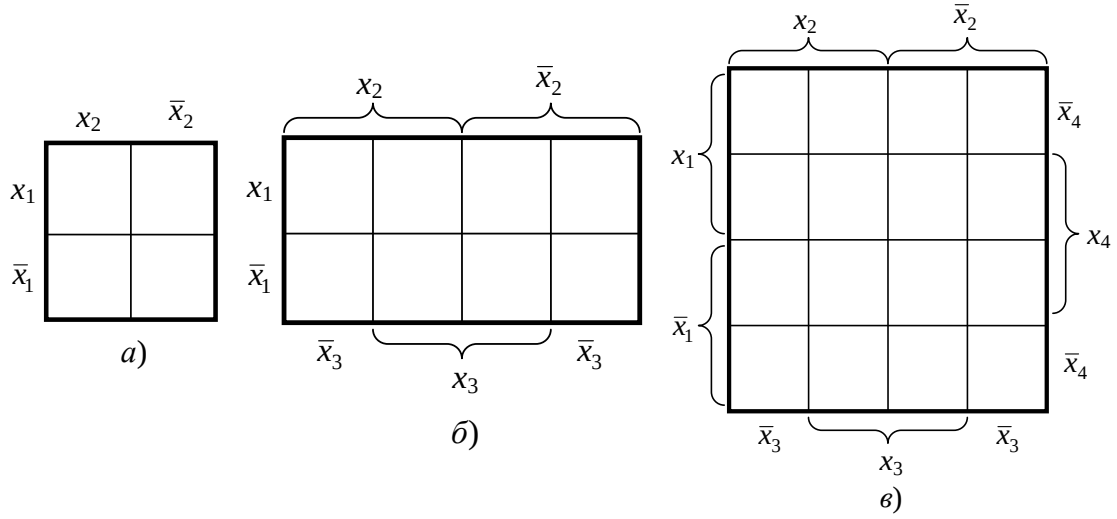


Рисунок 1.3 – Карты Вейча для функций нескольких переменных:
а) две переменные; *б*) три переменные; *в*) четыре переменные

Число клеток карты равно числу всех возможных наборов значений аргументов 2^n (n – число аргументов функций). В каждую из клеток карты записывается значение функции на соответствующем этой клетке наборе значений аргументов. Пусть логическая функция задана в виде таблицы истинности (таблица 1.2).

Таблица 1.2 – Таблица истинности логической функции трех аргументов

X_3	X_2	X_1	$F(X_3, X_2, X_1)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Тогда таблица истинности этой функции в форме карты Вейча может быть представлена рисунком 1.4.

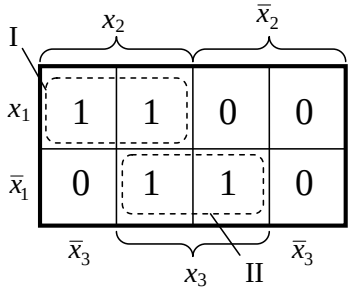


Рисунок 1.4 – Карта Вейча для логической функции трех переменных

Карта Вейча определяет значения функции на всех возможных наборах значений аргументов и является таблицей истинности. Карты Вейча компактны, но главное их достоинство состоит в следующем. При любом переходе из одной клетки в соседнюю вдоль столбца или строки изменяется значение лишь одного аргумента функции. Следовательно, если в паре соседних клеток содержится *1*, то над соответствующими им членами канонической формы может быть проведена операция склеивания. Таким образом, облегчается поиск склеиваемых членов.

Сформулируем правила получения МДНФ функций с помощью карт Вейча.

Все клетки, содержащие *1*, объединяются в замкнутые области. При этом каждая область должна представлять собой прямоугольник с числом клеток 2^k, где $k = 0, 1, 2, \ldots$ Значит, допустимое число клеток в области 1, 2, 4, 8, … Области могут пересекаться, и одни и те же клетки могут входить в разные области. Затем проводится запись выражения МДНФ функции. Каждая из областей в МДНФ представляется членом, число букв в котором на k меньше общего числа аргументов функции n (т.е. равно $n – k$). Каждый член МДНФ составляется лишь из тех аргументов, которые для клеток соответствующей области имеют одинаковое значение (без инверсии, либо с инверсией).

Таким образом, при охвате клеток замкнутыми областями следует стремиться, чтобы число областей было минимальным (при этом минимальным будет число членов в МДНФ функции), а каждая область содержала возможно большее число клеток (при этом минимальным будет число букв в членах МДНФ функции).

Рассмотрим минимизацию с помощью карты Вейча функции трех аргументов, представленной в таблице 1.2. Все клетки, содержащие *1*, охватываются двумя областями. В каждой из областей 2^1 клеток, для них $n - k = 3 - 1 = 2$, и эти области в МДНФ будут представлены членами, содержащими по две буквы. Первой области соответствует член $x_2 \cdot x_1$ (аргумент x_3 здесь не присутствует, так как для одной клетки этой области он имеет значение без инверсии, для другой – с инверсией). Второй области соответствует член $x_3 \cdot \bar{x}_1$. Следовательно, МДНФ функции:

$$f(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot \bar{x}_1.$$

При охвате клеток замкнутыми областями допускается сворачивание в цилиндр карты Вейча с объединением ее противоположных вертикальных сторон. В силу этого крайние клетки столбца таблицы рассматриваются как соседние и могут быть объединены в общую область. Карту Вейча функции четырех переменных можно представить объемной фигурой тора. Для такой карты допустимо сворачивание с объединением и вертикальных и горизонтальных сторон исходной развертки.

При построении логических устройств обычно не пользуются функционально полной системой логических элементов, реализующих все три основные логические операции: И, ИЛИ и НЕ. На практике, с целью сокращения номенклатуры элементов пользуются функционально полной системой элементов, включающей только два элемента, выполняющих операции И-НЕ и ИЛИ-НЕ, или даже только один из этих элементов. Причем число входов этих элементов, как правило, задано. Поэтому вопросы синтеза логических устройств в заданном базисе логических элементов имеют большое практическое значение.

Любую логическую функцию можно записать в требуемом базисе логических элементов. При этом используются два технических приема: двойное инвертирование исходного выражения или его части и применение теорем Де-Моргана.

Если требуется привести логическую функцию к базису логических элементов И-НЕ, то указанными приемами функция преобразуется к виду, содержащему только операции логического умножения и инверсии. Далее она переписывается через условные обозначения операции И-НЕ. Аналогично поступают при преобразовании логической функции к базису логических элементов ИЛИ-НЕ. В этом случае в выражении оставляют только операцию логического сложения и инверсии.

Рассмотрим последовательность синтеза логического устройства в заданном базисе на примере функции $f(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot \bar{x}_1$, полученной ранее как МДНФ.

Для перехода от базиса И, ИЛИ, НЕ, в котором представлено логическое выражение, к базису И-НЕ выполняются следующие действия:

- дважды инвертируется правая часть выражения:

$$f(x_3, x_2, x_1) = \overline{\overline{x_2 \cdot x_1 + x_3 \cdot \bar{x}_1}};$$

- проводится преобразование по теореме де Моргана:

$$f(x_3, x_2, x_1) = \overline{\overline{x_2 \cdot x_1} \cdot \overline{x_3 \cdot \bar{x}_1}}. \qquad (1.3)$$

Для перехода к базису ИЛИ-НЕ выполняются следующие действия:

- инвертируются обе части выражения:

$$\overline{f(x_3, x_2, x_1)} = \overline{x_2 \cdot x_1 + x_3 \cdot \bar{x}_1};$$

- каждый дизъюнктивный член выражения дважды инвертируются:

$$\overline{f(x_3, x_2, x_1)} = \overline{\overline{x_2 \cdot x_1} + \overline{x_3 \cdot \bar{x}_1}};$$

- проводится преобразование по теореме де Моргана:

$$\overline{f(x_3, x_2, x_1)} = \overline{\overline{\bar{x}_2 + \bar{x}_1} + \overline{\bar{x}_3 + \bar{\bar{x}}_1}} = \overline{\overline{\bar{x}_2 + \bar{x}_1} + \overline{\bar{x}_3 + x_1}};$$

- для получения неинвертированного значения функции *f* на выходе устройства добавляется инвертор в базисе ИЛИ-НЕ:

$$f(x_3, x_2, x_1) = \overline{\overline{\overline{x}_2 + \overline{x}_1} + \overline{\overline{x}_3 + x_1}}. \qquad (1.4)$$

Прием уменьшения фактического числа входов логического элемента базируется на использовании тождества булевой алгебры $x + x = x$ или $x \cdot x = x$, поэтому на несколько входов логического элемента можно подавать одну и ту же логическую переменную. Следствием сказанного является важный практический вывод: если на все входы *n*-входового элемента И-НЕ или ИЛИ-НЕ подать один и тот же логический сигнал, то относительно этого сигнала элемент превращается в инвертор.

Построенные в соответствии с (1.3) и (1.4) схемы логических устройств приведены на рисунке 1.5.

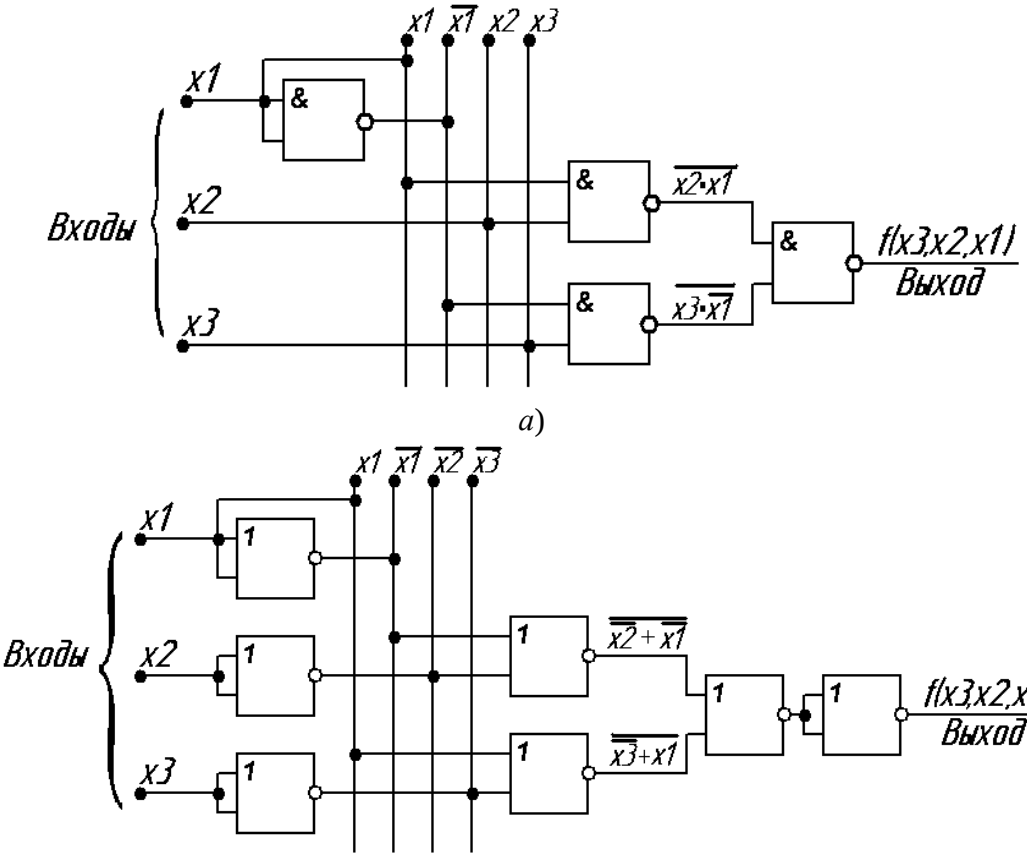


Рисунок 1.5 – Логические схемы устройств в базисе И-НЕ (*а*), ИЛИ-НЕ (*б*)

1.4 Пример минимизации логической функции и перехода в заданный базис

Логическая функция трех аргументов $f(x_3, x_2, x_1)$ задана в виде таблицы истинности (таблица 1.3).

Требуется для данной функции:

- построить структурную схему логического устройства на основе совершенной дизъюнктивной нормальной формы;

- построить структурную схему логического устройства на основе минимальной дизъюнктивной нормальной формы;

- построить структурную схему логического устройства в базисе логических элементов И-НЕ;

- построить структурную схему логического устройства в базисе логических элементов ИЛИ-НЕ.

Правильность построений подтвердить результатами моделирования в программе MicroCAP.

Таблица 1.3 – Таблица истинности логической функции

X_3	X_2	X_1	$F(X_3, X_2, X_1)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

I этап. Синтез логического устройства на основе СДНФ.

При выполнении этапа исследования использованы приемы №2, 3, 4, 5, 6 раздела «Типовые приемы работы в MicroCAP…».

Каждый из наборов аргументов, при которых логическая функция $f(x_3, x_2, x_1)$ равна единице (4, 6, 7, 8 строки таблицы истинности), является дизъюнктивным членом СДНФ. Следовательно, совершенная дизъюнктивная нормальная форма имеет вид:

$$f_{\text{СДНФ}}(x_3, x_2, x_1) = \overline{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot \overline{x}_2 \cdot x_1 + x_3 \cdot x_2 \cdot \overline{x}_1 + x_3 \cdot x_2 \cdot x_1.$$

Аппаратная реализация функции $f(x_3, x_2, x_1)$ в СДНФ потребует три инвертора (для выработки сигналов $\overline{x}_3, \overline{x}_2, \overline{x}_1$); четыре элемента 3И; один элемент 4ИЛИ. Для построения логической схемы необходимо логические элементы, предназначенные для выполнения операций, располагать от входа в порядке, определенном булевым выражением.

Структурная схема логического устройства на основе СДНФ, подготовленная в программе MicroCAP, представлена на рисунке 1.6.

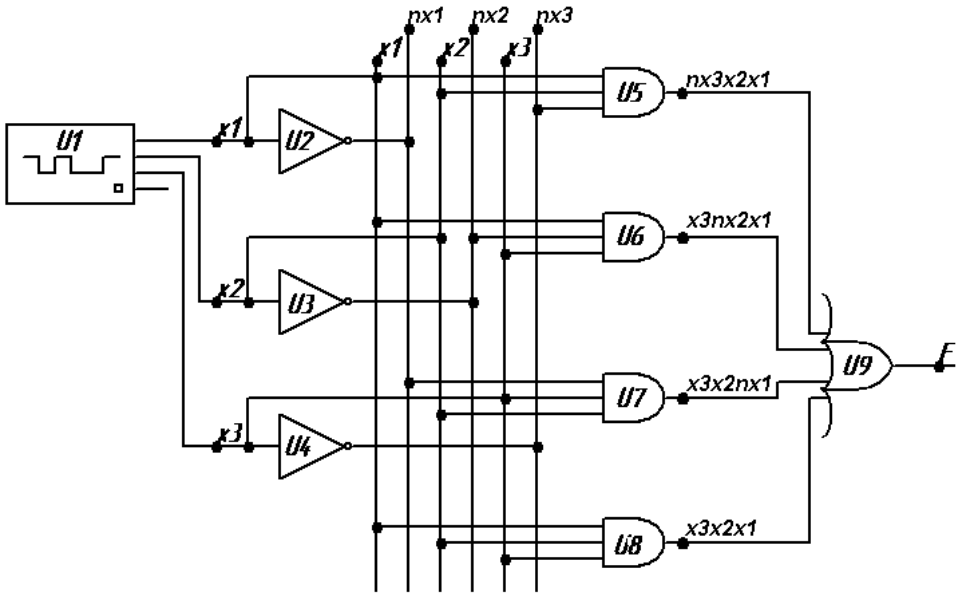


Рисунок 1.6 – Структурная схема логического устройства на основе СДНФ

Схема, предназначенная для моделирования в MicroCAP, дополнена четырехразрядным источником цифрового сигнала *U*1. Трехразрядных источников сигнала, как это требует наша схема, в MicroCAP не предусмотрено. Если число аргументов данной функции равно *n* = 3, то число различных сочетаний (наборов) аргументов составляет $2^n = 8$. Следовательно, для воспроизведения в MicroCAP всех сочетаний аргументов потребуется 8 тактов (циклов). Предположим, что длительность одного такта составляет *t* = 1 мкс, тогда общая продолжительность анализа по времени будет $\sum_{i=1}^{8} t_i = 8$ мкс. В нашем примере старший вывод источника *U*1 не используется, поскольку область определения логической функции находится в диапазоне $[0; 7]_{16}$.

В диалоговом окне свойств Stim 4 для источника *U*1 указывают информацию о выбранной системе счисления и о параметрах цифрового сигнала. В частности, в строке FORMAT указывают значение 4. Задать параметры цифрового сигнала для строки COMMAND можно несколькими способами. Рассмотрим наиболее простой способ – написание бесконечного программного цикла увеличения цифрового слова на единицу в каждом такте. При достижении максимального значения в выбранной системе счисления (число *F*) в следующем такте происходит обнуление. Листинг такой программы приведен ниже.

```
.DEFINE INPUT
+ 0us 0
+ LABEL begin
+ +1us INCR BY 1
+ +1us GOTO begin -1 TIMES
```

Правильность задания параметров цифрового сигнала оперативно проверим нажатием на кнопку Plot в диалоговом окне Stim 4 (рисунок 1.7). Временная зависимость разрядов D(2), D(3), D(4) представляет собой изменение трехразрядного слова согласно кодовым комбинациям таблицы истинности (таблица 1.3): 000, 001, 010, 011 и т.д.

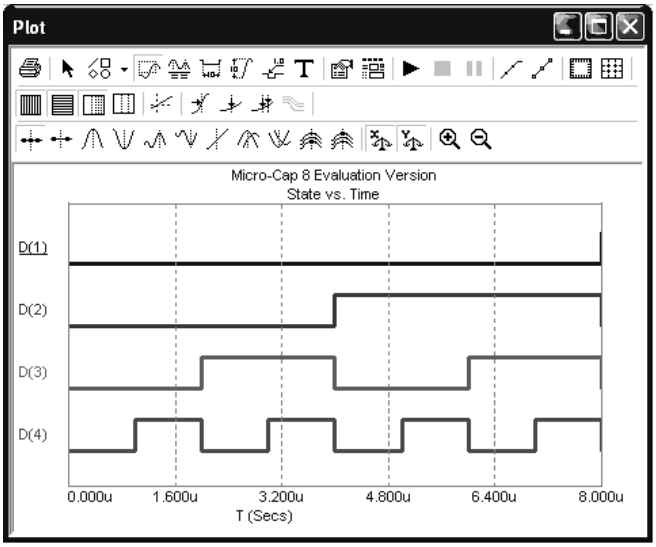


Рисунок 1.7 – Временная диаграмма цифрового сигнала

В диалоговых окнах свойств логических элементов НЕ, 3И, 4 ИЛИ в строке TIMING MODEL необходимо выбрать из списка справа имя D0_GATE. Выбранное имя соответствует идеальной модели динамики распространения цифрового сигнала – с нулевыми временами задержки.

На рисунке 1.6 входные зажимы электрической схемы обозначены как *x*1, *x*2, *x*3, а выход схемы – как *F*. В программе MicroCAP не предусмотрен символ инверсии. По этой при-

чине пользователь должен самостоятельно придумать символьное обозначение инвертированных сигналов своей схемы. В нашем примере для именования инвертированных сигналов применена приставка *n*. Например, имя *nx*1 на схеме означает инверсию сигнала *x*1, т.е. $\overline{x}_1$.

Моделирование во временной области проводится по команде *Analysis/Transient…* В появляющемся диалоговом окне Transient Analysis Limits указывают:

- в строке ввода Time Range 8u – общая продолжительность временного вида анализа;
- нажатием на кнопку Add устанавливают четыре табличные строки в нижней части диалогового окна;
- в столбце P всех строк таблицы записывается 1 – первый и единственный номер окна графика;
- в столбце X Expression всех строк таблицы записывается переменная T – резервированная переменная для обозначения времени;
- в столбце Y Expression в каждой из строк записывается по одному из выражений d(x1), d(x2), d(x3), d(f).

Опцию Auto Scale Range рекомендуется включить.

Нажатие на кнопку Run в диалоговом окне приводит к построению временной диаграммы работы устройства (рисунок 1.8). Нижняя зависимость *d*(*f*) на графике представляет собой значения логической функции. Сравнивая их с таблицей истинности (таблица 1.3, правый столбец) приходим к выводу об адекватности проведенного моделирования.

II этап. Синтез логического устройства на основе МДНФ.

При выполнении этапа исследования использованы приемы №2, 3, 4, 5, 6 раздела «Типовые приемы работы в MicroCAP…».

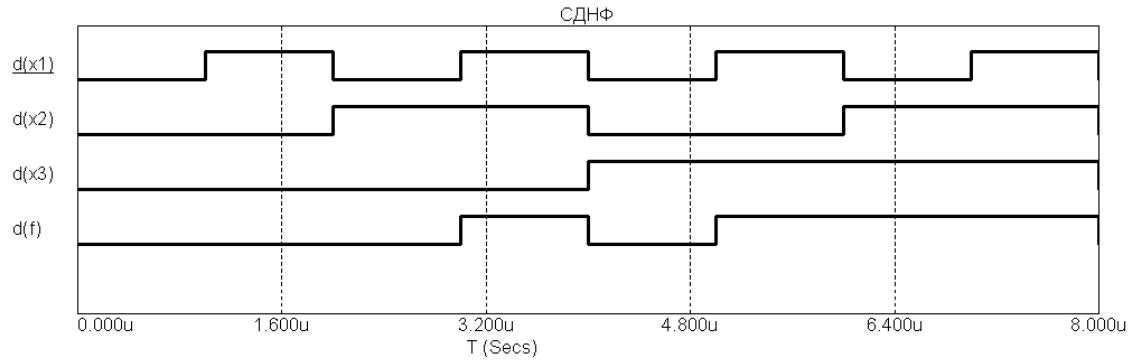


Рисунок 1.8 – Временная диаграмма работы схемы в СДНФ

Для проведения второго этапа исследования воспользуемся логической функцией в СДНФ, полученной на I этапе:

$$f_{\text{СДНФ}}(x_3, x_2, x_1) = \overline{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot \overline{x}_2 \cdot x_1 + x_3 \cdot x_2 \cdot \overline{x}_1 + x_3 \cdot x_2 \cdot x_1.$$

Графический образ такой логической функции представлен в виде куба на рисунке 1.9.

Множество нулевых кубов образуют нулевой кубический комплекс K_0 = (011, 101, 110, 111). Множество единичных кубов образуют единичный кубический комплекс K_1 = (-11, 1-1, 11-). Двоичного кубического комплекса для данной функции не существует, поскольку нет ни одной грани, полностью образованной единичными кубами.

Кубический комплекс логической функции имеет вид:

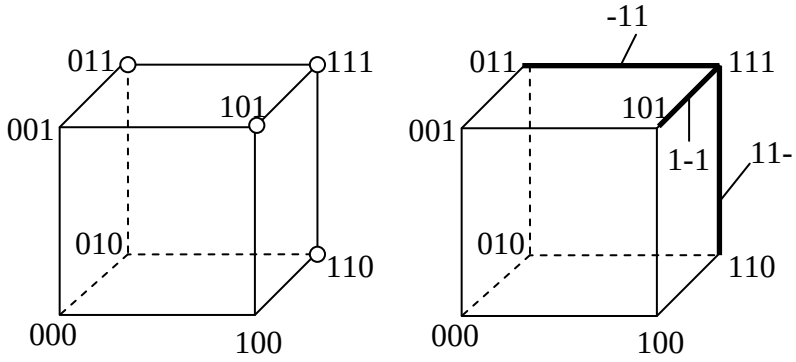
K(*f*) = (011, 101, 110, 111, -11, 1-1, 11-).

Нулевой кубический комплекс (рисунок 1.9, *а*) включает все вершины куба, поэтому образует покрытие функции:

$\Pi_1(z) = K_0$ = (011, 101, 110, 111).

Все вершины куба включаются также в единичный кубический комплекс K_1 (рисунок 1.9, *б*), поэтому он также образует покрытие функции:

$$П_2(z) = K_1 = (-11, 1\text{-}1, 11\text{-}).$$



а) *б*)

Рисунок 1.9 – Геометрическое представление логической функции:
а) нулевой кубический комплекс; *б*) единичный кубический комплекс

Перебирая сочетания кубов различных рангов можно получить следующие покрытия функции:

$$П_3(z) = (011, 1\text{-}1, 11\text{-});$$
$$П_4(z) = (101, -11, 11\text{-});$$
$$П_5(z) = (110, -11, 1\text{-}1);$$
$$П_6(z) = (111, -11, 1\text{-}1, 11\text{-}) \text{ и так далее.}$$

Среди перечисленных выше (и существующих вообще) покрытий выражение $П_2(z) = K_1 = (-11, 1\text{-}1, 11\text{-})$ имеет минимальную цену. Докажем это утверждение по формулам:

$$Ц_{Ki} = n - r;\ Ц_П = \Sigma\ Ц_{Ki},$$

где n – количество переменных логической функции; r – количество прочерков в записи i-ого куба (ранг куба).

Для покрытия $П_2(z)$ имеем:

$$Ц_{K1} = 3 - 1 = 2;\ Ц_{K2} = 3 - 1 = 2;\ Ц_{K3} = 3 - 1 = 2;\ Ц_П = \Sigma\ Ц_{Ki} = 2 + 2 + 2 = 6.$$

Все остальные покрытия имеют бóльшую цену. Следовательно, логическая функция $f_{МДНФ}(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2$, соответствующая покрытию $П_2(z)$, является минимальной дизъюнктивной нормальной формой (МДНФ).

Проверим полученный результат другим способом – минимизацией с помощью карты Вейча. Карта Вейча для исходной функции в СДНФ представлена на рисунке 1.10. В каждую из клеток карты записывается значение функции на соответствующем этой клетке наборе значений аргументов.

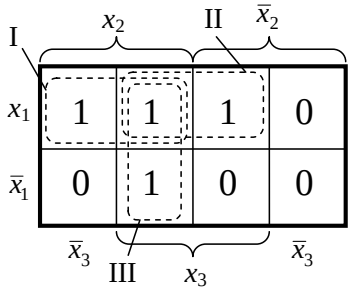


Рисунок 1.10 – Карта Вейча для исходной функции в СДНФ

Все клетки, содержащие единицы, охватываются тремя прямоугольными областями. В каждой из областей 2^1 клеток. Поскольку области могут пересекаться, то одни и те же

клетки входят в разные области. Первой области соответствует член $x_2 \cdot x_1$; второй области – $x_3 \cdot x_1$; третьей области – $x_3 \cdot x_2$. Следовательно, МДНФ функции:

$$f_{\text{МДНФ}}(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2 .$$

Напомним, что при охвате клеток замкнутыми областями следует стремиться, чтобы число областей было минимальным (при этом минимальным будет число членов в МДНФ функции), а каждая область содержала возможно большее число клеток (при этом минимальным будет число букв в членах МДНФ функции).

Аппаратная реализация логической функции в МДНФ потребует три элемента 2И и один элемент 3ИЛИ. Структурная схема логического устройства на основе МДНФ, подготовленная в программе MicroCAP, представлена на рисунке 1.11.

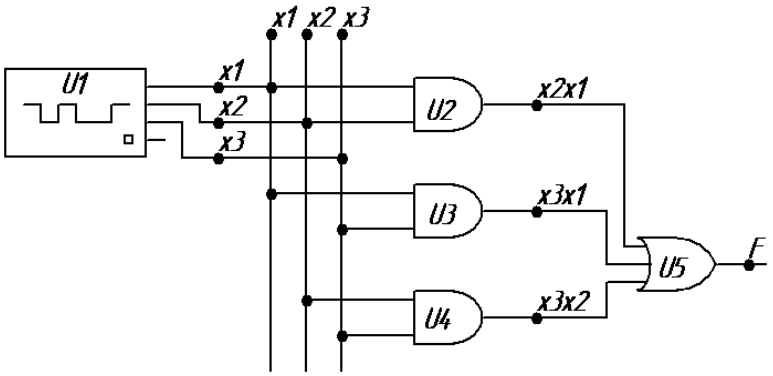


Рисунок 1.11 - Структурная схема логического устройства на основе МДНФ

Временной анализ схемы в MicroCAP проводится аналогично I этапу; результаты анализа представлены на рисунке 1.12. Сравнивая полученные временные зависимости с результатами предыдущего этапа, приходим к выводу об адекватности проведенного моделирования.

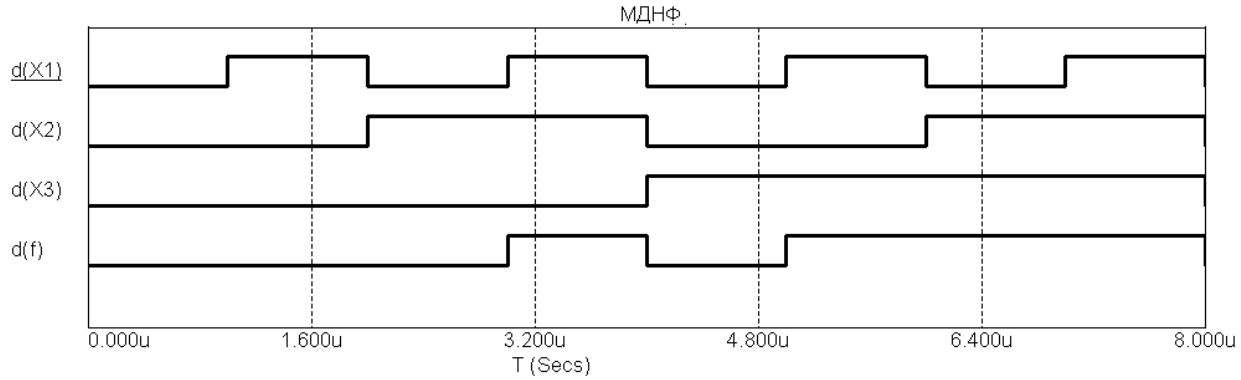


Рисунок 1.12– Временная диаграмма работы схемы в МДНФ

III этап. Синтез логического устройства в базисе И-НЕ.

При выполнении этапа исследования использованы приемы №2, 3, 4, 5, 6 раздела «Типовые приемы работы в MicroCAP…».

Для проведения третьего этапа исследования воспользуемся логической функцией в МДНФ, полученной на II этапе:

$$f_{\text{МДНФ}}(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2 .$$

Если требуется привести логическую функцию к базису логических элементов И-НЕ, то специальными приемами функция преобразуется к виду, содержащему только операции логического умножения и инверсии. Для перехода к базису И-НЕ выполняются следующие действия:

- дважды инвертируется правая часть выражения:

$$f(x_3, x_2, x_1) = \overline{\overline{x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2}};$$

- проводится преобразование по теореме де Моргана:

$$f(x_3, x_2, x_1) = \overline{\overline{x_2 \cdot x_1} \cdot \overline{x_3 \cdot x_1} \cdot \overline{x_3 \cdot x_2}}.$$

Следовательно, преобразованная логическая функция в базисе логических элементов И-НЕ выглядит как:

$$f_{\text{И-НЕ}}(x_3, x_2, x_1) = \overline{\overline{x_2 \cdot x_1} \cdot \overline{x_3 \cdot x_1} \cdot \overline{x_3 \cdot x_2}}.$$

Аппаратная реализация такой логической функции потребует три элемента 2И-НЕ и один элемент 3И-НЕ. Структурная схема логического устройства в базисе И-НЕ, подготовленная в программе MicroCAP, представлена на рисунке 1.13.

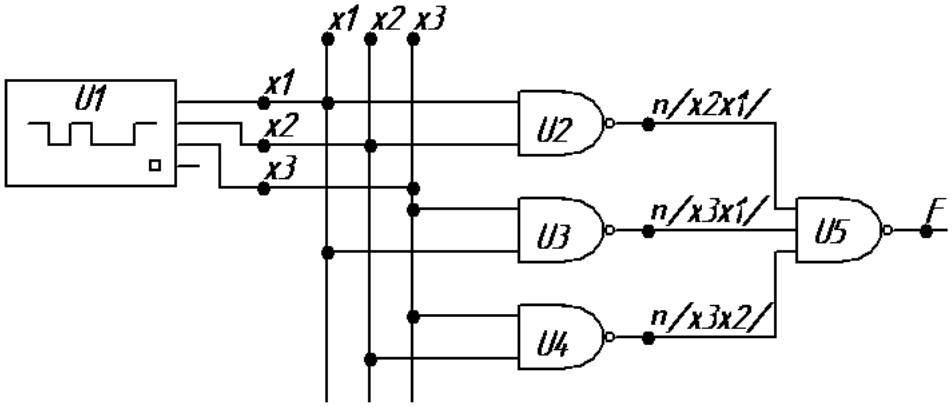


Рисунок 1.13 - Структурная схема логического устройства в базисе И-НЕ

Временной анализ схемы в MicroCAP проводится аналогично I этапу; результаты анализа представлены на рисунке 1.14. Сравнивая полученные временные зависимости с результатами предыдущего этапа, приходим к выводу об адекватности проведенного моделирования.

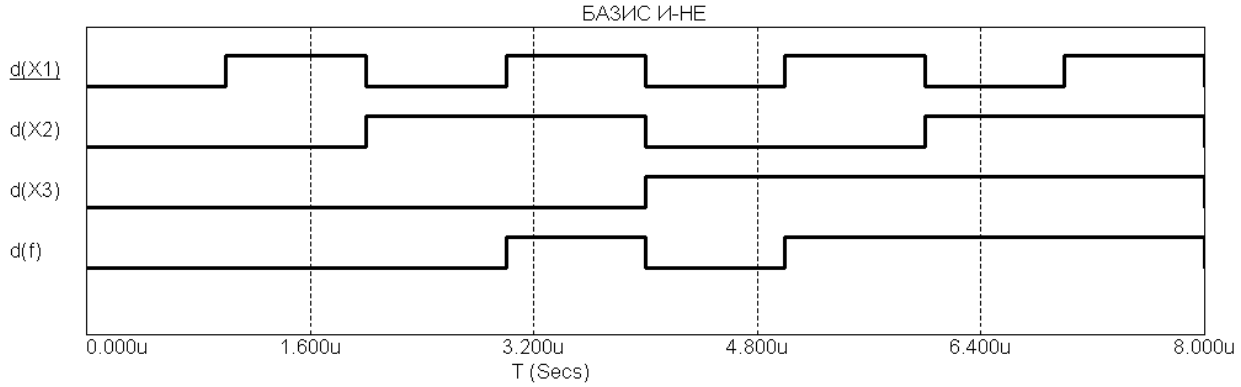


Рисунок 1.14 – Временная диаграмма работы схемы в базисе И-НЕ

IV этап. Синтез логического устройства в базисе ИЛИ-НЕ.

При выполнении этапа исследования использованы приемы №2, 3, 4, 5, 6 раздела «Типовые приемы работы в MicroCAP…».

Для проведения четвертого этапа исследования воспользуемся логической функцией в МДНФ, полученной на II этапе:

$$f_{\text{МДНФ}}(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2.$$

Если требуется привести логическую функцию к базису логических элементов ИЛИ-НЕ, то специальными приемами функция преобразуется к виду, содержащему только опера-

ции логического сложения и инверсии. Для перехода к базису ИЛИ-НЕ выполняются следующие действия:

- инвертируются обе части выражения:

$$\overline{f(x_3, x_2, x_1)} = \overline{x_2 \cdot x_1 + x_3 \cdot x_1 + x_3 \cdot x_2};$$

- каждый дизъюнктивный член выражения дважды инвертируются:

$$\overline{f(x_3, x_2, x_1)} = \overline{\overline{x_2 \cdot x_1} + \overline{x_3 \cdot x_1} + \overline{x_3 \cdot x_2}};$$

- проводится преобразование по теореме де Моргана:

$$\overline{f(x_3, x_2, x_1)} = \overline{\overline{\overline{x}_2 + \overline{x}_1} + \overline{\overline{x}_3 + \overline{x}_1} + \overline{\overline{x}_3 + \overline{x}_2}};$$

- для получения неинвертированного значения функции f на выходе устройства добавляется инвертор в базисе ИЛИ-НЕ:

$$f(x_3, x_2, x_1) = \overline{\overline{\overline{x}_2 + \overline{x}_1} + \overline{\overline{x}_3 + \overline{x}_1} + \overline{\overline{x}_3 + \overline{x}_2}}.$$

Таким образом, преобразованная логическая функция в базисе логических элементов ИЛИ-НЕ выглядит как:

$$f_{\text{ИЛИ-НЕ}}(x_3, x_2, x_1) = \overline{\overline{\overline{x}_2 + \overline{x}_1} + \overline{\overline{x}_3 + \overline{x}_1} + \overline{\overline{x}_3 + \overline{x}_2}}.$$

Аппаратная реализация такой логической функции потребует семь элемента 2ИЛИ-НЕ, среди которых четыре элемента нужны для выработки инверсных сигналов ($\overline{x}_1, \overline{x}_2, \overline{x}_3, f$), и один элемент 3ИЛИ-НЕ. Напомним, что если на все входы n-входового элемента ИЛИ-НЕ подать один и тот же логический сигнал, то относительно этого сигнала элемент превращается в инвертор. Структурная схема логического устройства в базисе ИЛИ-НЕ, подготовленная в программе MicroCAP, представлена на рисунке 1.15.

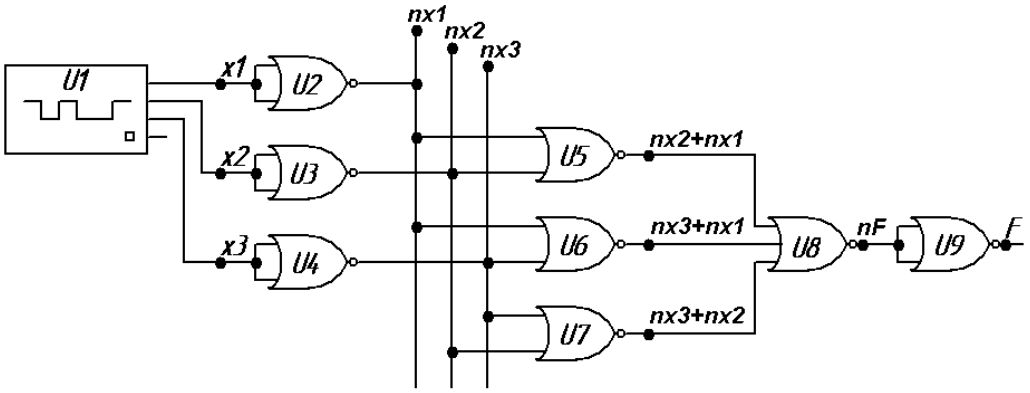


Рисунок 1.15 - Структурная схема
логического устройства в базисе ИЛИ-НЕ

Временной анализ схемы в MicroCAP проводится аналогично I этапу; результаты анализа представлены на рисунке 1.16. Сравнивая полученные временные зависимости с результатами предыдущего этапа, приходим к выводу об адекватности проведенного моделирования.

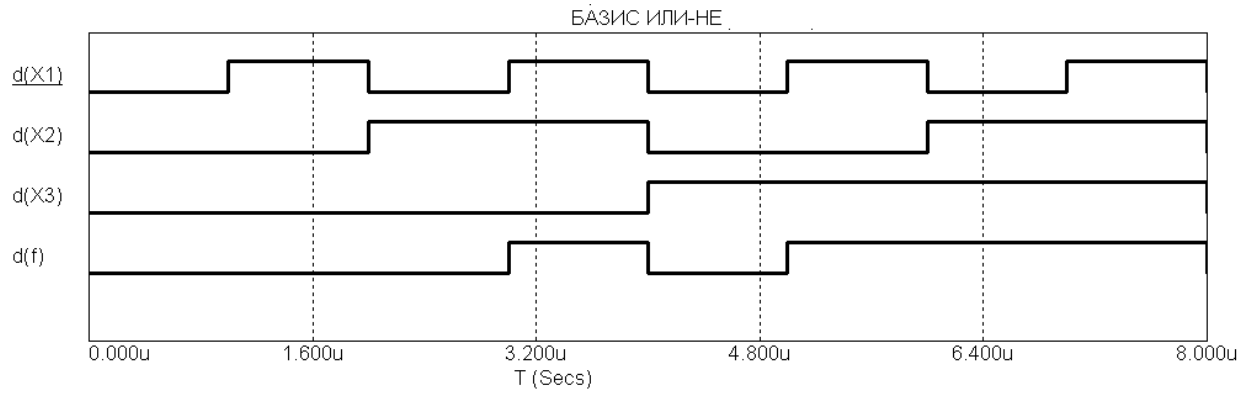


Рисунок 1.16 - Временная диаграмма работы схемы в базисе ИЛИ-НЕ

1.5 Лабораторное задание

Повторить методический пример, приведенный выше, по исходным данным Вашего варианта. *Критерием правильности* проведенного исследования является совпадение временных зависимостей выходного сигнала для всех схем. Временная зависимость выходного сигнала, в свою очередь, должна повторять значения логической функции в таблице истинности.

1.6 Контрольные вопросы

1. В чем заключаются этапы синтеза логического устройства?
2. Что такое дизъюнктивная нормальная форма?
3. Как происходит переход от дизъюнктивной нормальной формы к совершенной дизъюнктивной нормальной форме?
4. Каково правило записи совершенной дизъюнктивной нормальной формы для функции, заданной таблично?
5. В чем недостаток структурных схем, построенных по СДНФ?
6. Что такое нулевой, единичный, двоичный куб?
7. Что такое ранг куба?
8. Что такое кубический комплекс?
9. Что такое покрытие логической функции?
10. Что такое цена покрытия логической функции?
11. Что такое карта Вейча?

1.7 Варианты заданий

Для всех вариантов задания общими являются три столбца таблицы истинности (таблица 1.4), которые представляют собой восемь возможных сочетаний аргументов логической функции $f(x_3, x_2, x_1)$. Варианты значений логической функции приведены в таблице 1.5. Выбрав соответствующий своему варианту столбец в таблице 1.5, необходимо присоединить его в качестве четвертого столбца к таблице 1.4 и получить, таким образом, полноценную таблицу истинности.

Таблица 1.4 – Сочетания аргументов логической функции

X_3	X_2	X_1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Таблица 1.5 – Варианты значений логической функции $f(x_3, x_2, x_1)$

	Номера вариантов												
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*
Значения функции	1	1	1	1	1	1	1	1	0	1	0	0	0
	0	1	1	1	0	0	0	0	0	0	1	1	0
	1	0	1	1	1	1	1	1	1	0	1	0	1
	0	0	0	1	0	0	0	0	1	1	0	1	0
	1	1	0	0	1	0	1	0	1	1	1	1	0
	0	0	0	0	1	0	0	1	0	0	0	0	1
	1	1	1	0	0	1	0	1	1	1	1	1	1
	0	0	0	0	0	1	1	0	0	0	0	0	1
	Номера вариантов												
	14	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	
Значения функции	0	1	0	0	0	0	1	1	0	1	1	1	
	1	0	0	0	0	1	1	0	0	1	0	1	
	0	0	0	1	0	0	0	1	1	1	0	1	
	0	0	0	1	1	1	1	0	1	0	1	0	
	0	0	1	1	1	0	0	1	1	1	0	0	
	1	1	1	1	1	1	1	0	1	0	1	0	
	1	1	1	0	1	0	0	1	1	0	1	1	
	1	1	1	0	0	1	1	1	0	1	1	1	

2 ЛАБОРАТОРНАЯ РАБОТА №2 – ОСОБЫЕ СЛУЧАИ СИНТЕЗА КОМБИНАЦИОННЫХ ЛОГИЧЕСКИХ УСТРОЙСТВ

2.1 Цель работы

В ходе выполнения настоящей работы предусматривается:

1) знакомство с процессом минимизации логических функций по методу Квайна;

2) изучение принципов доопределения логических функций с последующим нахождением минимальной формы;

3) изучение принципов минимизации систем логических функций на основе импликантных матриц;

4) приобретение практических навыков по использованию операций склеивания и поглощения.

2.2 Порядок выполнения работы

1. Изучить методические указания к лабораторной работе.
2. Письменно, в отчете по лабораторной работе ответить на контрольные вопросы.
3. Внимательно ознакомиться с примером, приведенным в пункте 2.4.
4. Выполнить лабораторное задание согласно варианту задания.
5. Сделать выводы по работе.

Внимание! Отчет по лабораторной работе в обязательном порядке должен содержать: схемы включения, графики зависимостей, все необходимые расчеты и их результаты, текстовые пояснения. На графиках в отчете должны присутствовать единицы измерения, масштаб, цена деления.

2.3 Синтез недоопределенных логических функций. Синтез логических устройств с несколькими выходами

Для минимизации логических функций часто используются методы, обладающие однозначностью алгоритма, что является предпосылкой применения ЭВМ. К таким методам относится, в частности, метод Квайна. Метод Квайна позволяет представлять функции в дизъюнктивной нормальной форме с минимальным числом членов и минимальным числом букв в членах. Этот метод содержит два этапа преобразования выражения функции: на первом этапе осуществляется переход от канонической формы (СДНФ) к *сокращенной форме*, на втором этапе – переход от сокращенной формы логического выражения к *минимальной форме*.

Получение сокращенной формы. Пусть заданная функция f представлена в СДНФ. Переход к сокращенной форме основан на последовательном применении двух операций: *операции склеивания* и *операции поглощения*. Для выполнения операции склеивания в выражении функции выявляются пары членов вида $w \cdot x$ и $w \cdot \bar{x}$, различающихся лишь тем, что один из аргументов в одном из членов представлен без инверсии, а в другом – с инверсией. Затем проводится склеивание таких пар членов: $w \cdot x + w \cdot \bar{x} = w \cdot (x + \bar{x}) = w$, и результаты склеивания w вводятся в выражение функции в качестве дополнительных членов. Далее выполняется операция поглощения. Она основана на равенстве $w + w \cdot z = w \cdot (1 + z) = w$ (член w

поглощает член $w \cdot z$). При проведении этой операции из логического выражения вычеркиваются все члены, поглощаемые членами, которые введены в результате операции склеивания. Операции склеивания и поглощения выполняются последовательно до тех пор, пока это возможно.

Члены сокращенной формы называются *простыми импликантами* функции.

Получение минимальной формы. Сокращенная форма может содержать лишние члены, исключение которых из выражения не повлияет на значение функции. Дальнейшее упрощение логического выражения достигается исключением из выражения лишних членов. В этом заключается содержание второго этапа минимизации. Покажем этот этап минимизации логического выражения на примере построения логического устройства для функции, заданной в таблице 2.1.

Таблица 2.1 – Таблица истинности логической функции

X_4	X_3	X_2	X_1	$F(X_4,X_3,X_2,X_1)$
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

СДНФ этой функции:

$$f(x_4,x_3,x_2,x_1)=\overline{x}_4\cdot\overline{x}_3\cdot\overline{x}_2\cdot\overline{x}_1+\overline{x}_4\cdot x_3\cdot\overline{x}_2\cdot\overline{x}_1+\overline{x}_4\cdot x_3\cdot x_2\cdot\overline{x}_1+$$
$$+\overline{x}_4\cdot x_3\cdot x_2\cdot x_1+x_4\cdot\overline{x}_3\cdot\overline{x}_2\cdot\overline{x}_1+x_4\cdot x_3\cdot x_2\cdot x_1.$$

Для получения сокращенной формы проводим операции склеивания и поглощения, а также используем теорему булевой алгебры $a+F\cdot\overline{a}=a+F$:

$$f(x_4,x_3,x_2,x_1)=\overline{x}_3\cdot\overline{x}_2\cdot\overline{x}_1\cdot\left(\overline{x}_4+x_4\right)+\overline{x}_4\cdot x_3\cdot x_2\cdot\left(\overline{x}_1+x_1\right)+\overline{x}_4\cdot x_3\cdot\overline{x}_2\cdot\overline{x}_1+$$
$$+x_4\cdot x_3\cdot x_2\cdot x_1=\overline{x}_3\cdot\overline{x}_2\cdot\overline{x}_1+\overline{x}_4\cdot x_3\cdot x_2+\overline{x}_4\cdot x_3\cdot\overline{x}_2\cdot\overline{x}_1+x_4\cdot x_3\cdot x_2\cdot x_1=$$
$$=\overline{x}_3\cdot\overline{x}_2\cdot\overline{x}_1+\overline{x}_4\cdot x_3\cdot\left(x_2+\overline{x}_2\cdot\overline{x}_1\right)+x_4\cdot x_3\cdot x_2\cdot x_1=\overline{x}_3\cdot\overline{x}_2\cdot\overline{x}_1+\overline{x}_4\cdot x_3\cdot x_2+ \quad (2.1)$$
$$+\overline{x}_4\cdot x_3\cdot\overline{x}_1+x_4\cdot x_3\cdot x_2\cdot x_1=\overline{x}_3\cdot\overline{x}_2\cdot\overline{x}_1+x_3\cdot x_2\cdot\left(\overline{x}_4+x_4\cdot x_1\right)+\overline{x}_4\cdot x_3\cdot\overline{x}_1=$$
$$=\overline{x}_3\cdot\overline{x}_2\cdot\overline{x}_1+\overline{x}_4\cdot x_3\cdot x_2+x_3\cdot x_2\cdot x_1+\overline{x}_4\cdot x_3\cdot\overline{x}_1.$$

Выражение (2.1) представляет собой сокращенную форму логического выражения заданной функции, а члены его являются простыми импликантами функции. Переход от сокращенной формы к минимальной осуществляется с помощью импликантной матрицы, приведенной в таблице 2.2.

В столбцы импликантной матрицы вписываются члены СДНФ заданной функции, в строки – простые импликанты функции, т.е. члены сокращенной формы логического выражения функции. Отмечаются крестиками столбцы членов СДНФ, поглощаемых отдельными простыми импликантами.

Таблица 2.2 – Импликантная матрица

Простая импликанта	$\bar{x}_4\bar{x}_3\bar{x}_2\bar{x}_1$	$\bar{x}_4x_3\bar{x}_2\bar{x}_1$	$\bar{x}_4x_3x_2\bar{x}_1$	$\bar{x}_4x_3x_2x_1$	$x_4\bar{x}_3\bar{x}_2\bar{x}_1$	$x_4x_3x_2x_1$
$\bar{x}_3\bar{x}_2\bar{x}_1$	(×)				(×)	
$\bar{x}_4x_3x_2$			×	×		
$x_3x_2x_1$				(×)		(×)
$\bar{x}_4x_3\bar{x}_1$		(×)	(×)			

В приведенной матрице, например, простая импликанта $\bar{x}_4x_3\bar{x}_1$ поглощает члены $\bar{x}_4x_3\bar{x}_2\bar{x}_1$ и $\bar{x}_4x_3x_2\bar{x}_1$, поэтому во втором и третьем столбцах последней строки поставлены крестики. Импликанты, которые не могут быть лишними и, следовательно, не могут быть исключены из сокращенной формы, составляют *ядро*. Входящие в ядро импликанты легко определяются по импликантной матрице. Для каждой из них имеется хотя бы один столбец, перекрываемый только данной импликантой.

В рассматриваемом примере ядро составляют импликанты $\bar{x}_3\bar{x}_2\bar{x}_1$; $x_3x_2x_1$; $\bar{x}_4x_3\bar{x}_1$, потому что только ими перекрываются первый, второй, пятый и шестой столбцы матрицы. Ядро выделено в матрице скругленными прямоугольниками.

Для получения минимальной дизъюнктивной формы необходимо, в первую очередь, включить простые импликанты, составляющие ядро функции. Затем, достаточно выбрать из импликант, не входящих в ядро, такое минимальное их количество с минимальным количеством букв в каждой из этих импликант, которое обеспечит перекрытие всех столбцов, не перекрытых членами ядра.

В рассматриваемом примере не требуется включать дополнительные импликанты, поскольку ядро функции перекрывает все столбцы. Следовательно, МДНФ заданной функции выглядит как:

$$f_{\text{МДНФ}}(x_4, x_3, x_2, x_1) = \bar{x}_3 \cdot \bar{x}_2 \cdot \bar{x}_1 + x_3 \cdot x_2 \cdot x_1 + \bar{x}_4 \cdot x_3 \cdot \bar{x}_1 .$$

Синтез недоопределенных логических функций. По условиям работы логического устройства некоторые наборы значений аргументов могут оказаться запрещенными для данного устройства и никогда не появиться на его входах. В этом случае функция задана не на всех наборах аргументов. Такие функции будем называть *недоопределенными.*

При синтезе логического устройства, реализующего не полностью заданную функцию, допустимо задаваться произвольными значениями функции на запрещенных наборах аргументов. При этом в зависимости от способа задания этих значений функции минимальная форма может оказаться простой или более сложной. Таким образом, возникает проблема целесообразного доопределения функции на запрещенных наборах аргументов.

Может быть использован следующий способ получения минимальной формы не полностью заданной функции *f*:

а) записывается СДНФ функции f_0, полученной из функции *f* заданием значения *0* на всех запрещенных наборах аргументов;

б) записывается СДНФ функции f_1, полученной из функции *f* заданием значения *1* на всех запрещенных наборах аргументов;

в) функция f_1 приводится к сокращенной форме (к форме, содержащей все простые импликанты);

г) составляется импликантная таблица из всех членов функции f_0 и простых импликант функции f_1;

д) искомая минимальная форма составляется из простых импликант функции f_1, поглощающих все члены СДНФ функции f_0.

Рассмотрим применение данного метода к минимизации недоопределенной функции, приведенной в таблице 2.3.

Методом Квайна приводим функцию f_1 к сокращенной форме:

$$f_1(x_3, x_2, x_1) = \overline{x}_3 \cdot \overline{x}_2 \cdot \overline{x}_1 + \overline{x}_3 \cdot \overline{x}_2 \cdot x_1 + \overline{x}_3 \cdot x_2 \cdot \overline{x}_1 + \overline{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot \overline{x}_2 \cdot x_1 +$$

$$+x_3 \cdot x_2 \cdot \overline{x}_1 + x_3 \cdot x_2 \cdot x_1 = \overline{x}_3 \cdot \overline{x}_2 \cdot \overline{x}_1 + x_2 \cdot \overline{x}_1 \cdot (\overline{x}_3 + x_3) + \overline{x}_2 \cdot x_1 \cdot (\overline{x}_3 + x_3) +$$

$$+x_2 \cdot x_1 \cdot (\overline{x}_3 + x_3) = \overline{x}_3 \cdot \overline{x}_2 \cdot \overline{x}_1 + x_2 \cdot \overline{x}_1 + \overline{x}_2 \cdot x_1 + x_2 \cdot x_1 = \overline{x}_1 \cdot (\overline{x}_3 \cdot \overline{x}_2 + x_2) +$$

$$+x_1 \cdot (\overline{x}_2 + x_2) = \overline{x}_1 \cdot (\overline{x}_3 + x_2) + x_1 = \overline{x}_3 + x_2 + x_1.$$

Составляем импликантную матрицу (таблица 2.4).

Таблица 2.3 – Таблица истинности недоопределенной логической функции

X_3	X_2	X_1	$F(X_3, X_2, X_1)$
0	0	0	---
0	0	1	1
0	1	0	1
0	1	1	---
1	0	0	0
1	0	1	---
1	1	0	---
1	1	1	1

Таблица 2.4 – Импликантная матрица

Простые импликанты функции f_1	$\overline{x}_3\overline{x}_2x_1$	$\overline{x}_3x_2\overline{x}_1$	$x_3x_2x_1$
$\overline{x}_3$	×	×	
X_2		×	×
X_1	×		×

Минимальная форма логического выражения может быть получена тремя способами:

$$f(x_3, x_2, x_1) = \overline{x}_3 + x_2 \text{ или}$$

$$f(x_3, x_2, x_1) = x_2 + x_1 \text{ или}$$

$$f(x_3, x_2, x_1) = \overline{x}_3 + x_1.$$

Рассмотрим минимизацию той же функции методом карт Вейча (рисунок 2.1). При минимизации функции данным методом следует на запрещенных наборах аргументов задавать такие значения, при которых клетки со значением *1* охватываются минимальным числом областей с максимальным числом клеток в каждой области. Применительно к рассматриваемой функции такое доопределение может быть осуществлено тремя различными способами, представленными на рисунке 2.2. Они приводят к полученным выше выражениям МДНФ функции.

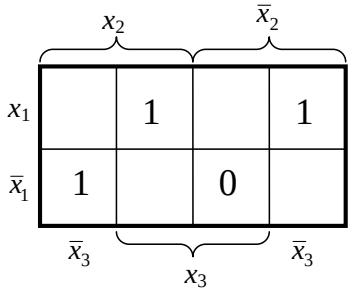
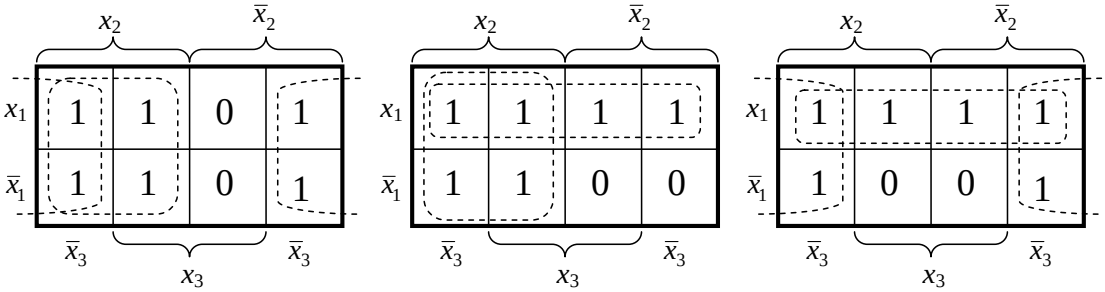


Рисунок 2.1 – Карта Вейча для недоопределенной логической функции



а) $\overline{x}_3 + x_2$; *б*) $x_2 + x_1$; *в*) $\overline{x}_3 + x_1$

Рисунок 2.2 – Различные способы доопределения логической функции

Синтез логических устройств с несколькими выходами. Пусть синтезируемое логическое устройство имеет *n* входов и *m* выходов (рисунок 2.3). На каждом из выходов должна быть сформирована определенная функция входных переменных. Эта задача могла бы быть решена синтезированием раздельно действующих узлов, каждый из которых реализовывал бы определенную выходную функцию. Однако, если даже каждый из этих узлов будет построен минимальным образом, в целом логическое устройство может оказаться не минимальным. Действительно, такое устройство могло бы быть минимизировано путем использования общих элементов в нескольких узлах, реализующих различные выходные функции.

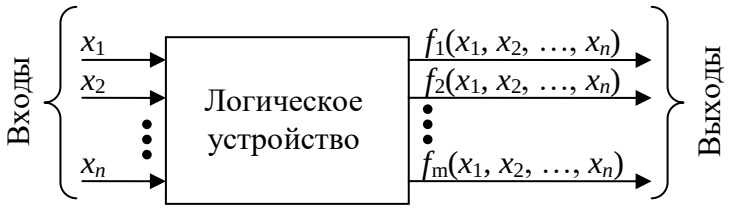


Рисунок 2.3 – Логическое устройство с несколькими выходами

Из этих соображений приведение каждой из выходных функций к минимальной форме не является условием получения минимального в целом устройства. При минимизации устройства в целом некоторые из функций могут оказаться представленными в неминимальной форме.

Принцип получения минимальной формы устройства сводится к нахождению минимального набора членов с минимальным числом входящих в них букв, достаточного для получения всех формируемых устройством функций.

Метод построения минимальных логических устройств с несколькими выходами рассмотрим на примере реализации устройства, способ функционирования которого задан в таблице 2.5.

Записываем в первый столбец таблицы 2.6 наборы аргументов, на которых хотя бы одна из выходных функций имеет значение *1*. Во второй столбец таблицы 2.6 в качестве признака записываем функции, принимающие значение *1* при данном наборе аргументов.

Таблица 2.5 – Таблица истинности логического устройства с тремя выходами

X_3	X_2	X_1	$F_1(X_3, X_2, X_1)$	$F_2(X_3, X_2, X_1)$	$F_3(X_3, X_2, X_1)$
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	0	0

Таблица 2.6 – Признаки наборов аргументов

Наборы аргументов	Признаки наличия единицы	Признак поглощения таблицей 2.7
$\bar{x}_3\bar{x}_2x_1$	f_2f_3	√
$\bar{x}_3x_2\bar{x}_1$	f_2f_3	
$\bar{x}_3x_2x_1$	f_1	√
$x_3\bar{x}_2\bar{x}_1$	f_1f_3	√
$x_3\bar{x}_2x_1$	f_2f_3	√
$x_3x_2\bar{x}_1$	f_1f_3	√
$x_3x_2x_1$	f_1	√

Далее проводится операция склеивания, и получающиеся при этом члены заносятся в таблицу 2.7, рядом с членами записываются признаки в виде функций, общих в признаках той пары членов таблицы 2.6, склеиванием которых они получены.

Таблица 2.7 – Признаки наборов аргументов после склеивания

Импликанты	Признаки
$\bar{x}_2 \cdot x_1$	$F_2\,F_3$
$x_2 \cdot \bar{x}_1$	F_3
$x_2 \cdot x_1$	F_1
$x_3 \cdot \bar{x}_2$	F_3
$x_3 \cdot \bar{x}_1$	F_1F_3
$x_3 \cdot x_2$	F_1

Так, склеивание членов таблицы 2.6 $x_3 \cdot \bar{x}_2 \cdot \bar{x}_1$ (f_1f_3) и $x_3 \cdot x_2 \cdot \bar{x}_1$ (f_1f_3) приводит в таблице 2.7 к члену $x_3 \cdot \bar{x}_1$ (f_1f_3); склеивание членов $x_3 \cdot \bar{x}_2 \cdot \bar{x}_1$ (f_1f_3) и $x_3 \cdot \bar{x}_2 \cdot x_1$ (f_2f_3) приводит к члену $x_3 \cdot \bar{x}_2$ (f_3) и т.д. Операция склеивания не проводится над членами, в признаках которых не имеется общих функций.

Далее реализуется операция поглощения членами таблицы 2.7 членов таблицы 2.6. *Операция поглощения может проводиться лишь над членами, имеющими одинаковую ком-*

бинацию функций в признаках. Указанные операции склеивания и поглощения повторяются до тех пор, пока это возможно. Так, в таблице 2.6 галочками отмечены члены, поглотившиеся в результате членами таблицы 2.7.

Затем составляется импликантная матрица (таблица 2.8). Поскольку один из членов таблицы 2.6 не поглотился, его записывают в таблицу 2.8 наряду с простыми импликантами. Определяется набор импликант, обеспечивающий перекрытие всех столбцов импликантной матрицы (таблица 2.9).

Таблица 2.8 – Импликантная матрица

Импликанты	$\bar{x}_3\bar{x}_2x_1$		$\bar{x}_3x_2\bar{x}_1$		$\bar{x}_3x_2x_1$	$x_3\bar{x}_2\bar{x}_1$		$x_3\bar{x}_2x_1$		$x_3x_2\bar{x}_1$		$x_3x_2x_1$
	f_2	f_3	f_2	f_3	f_1	f_1	f_3	f_2	f_3	f_1	f_3	f_1
$\bar{x}_2x_1\,(f_2f_3)$	×	×						×	×			
$x_2\bar{x}_1\,(f_3)$				×							×	
$x_2x_1\,(f_1)$					×							×
$x_3\bar{x}_2\,(f_3)$							×		×			
$x_3\bar{x}_1\,(f_1f_3)$						×	×			×	×	
$x_3x_2\,(f_1)$										×		×
$\bar{x}_3x_2\bar{x}_1\,(f_2f_3)$			×	×								

Таблица 2.9 – Набор импликант, обеспечивающих перекрытие

Импликанты	Признаки
$\bar{x}_2 \cdot x_1$	$F_2\,F_3$
$\bar{x}_3 \cdot x_2 \cdot \bar{x}_1$	$F_2\,F_3$
$x_2 \cdot x_1$	F_1
$x_3 \cdot \bar{x}_1$	$F_1\,F_3$

Записываем для выходных функций логические выражения, составленные из этих импликант, в признаках которых содержатся заданные функции:

$$f_1(x_3, x_2, x_1) = x_2 \cdot x_1 + x_3 \cdot \bar{x}_1;$$

$$f_2(x_3, x_2, x_1) = \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot \bar{x}_1;$$

$$f_3(x_3, x_2, x_1) = \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_1.$$

Легко убедиться, что выражение для функции $f_3(x_3, x_2, x_1)$ не является минимальным. Минимальная для этой функции форма:

$$f_3(x_3, x_2, x_1) = \bar{x}_2 \cdot x_1 + \bar{x}_3 \cdot x_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_1 =$$

$$= \bar{x}_1 \cdot (\bar{x}_3 \cdot x_2 + x_3) + \bar{x}_2 \cdot x_1 = x_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_1 + \bar{x}_2 \cdot x_1$$

невыгодна для всей системы из-за члена $x_2 \cdot \bar{x}_1$, так как он не присутствует в функциях $f_1(x_3, x_2, x_1)$ и $f_2(x_3, x_2, x_1)$.

На рисунке 2.4 приведена структурная схема логического устройства, обеспечивающего заданное таблицей 2.5 функционирование. Как видно из схемы, ряд элементов участвует в формировании нескольких функций.

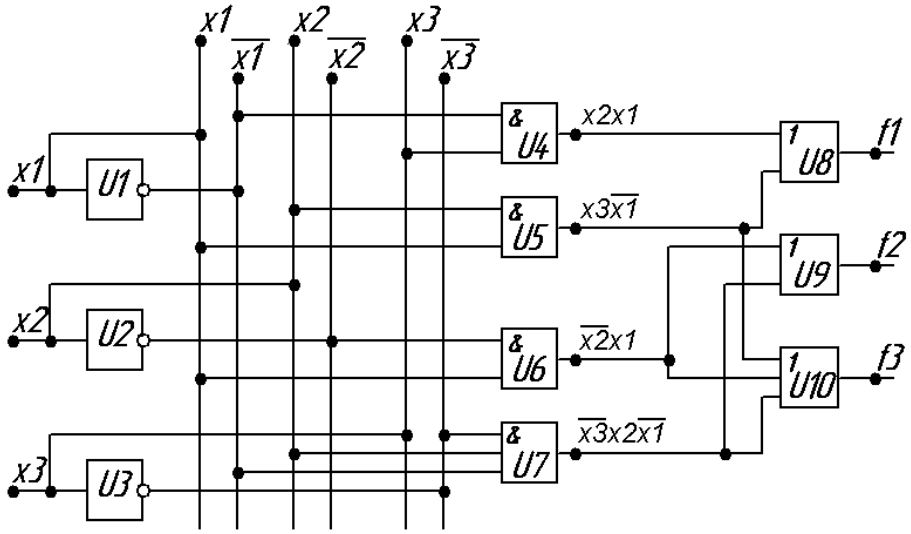


Рисунок 2.4 – Структурная схема логического устройства
с тремя выходами

2.4 Пример синтеза логических устройств

Недоопределенная логическая функция трех аргументов $f(x_3, x_2, x_1)$ и система логических функций трех аргументов

$$\begin{cases} g_1(x_3, x_2, x_1); \\ g_2(x_3, x_2, x_1); \\ g_3(x_3, x_2, x_1), \end{cases}$$

заданы в виде таблиц истинности (таблицы 2.10, 2.11).

Для логической функции $f(x_3, x_2, x_1)$ требуется:

- провести доопределение на запрещенных наборах аргументов с целью получения минимально возможной формы;
- построить структурную схему логического устройства в произвольном базисе по найденной минимальной форме.

Правильность построений подтвердить результатами моделирования в программе MicroCAP.

Для системы логических функций

$$\begin{cases} g_1(x_3, x_2, x_1); \\ g_2(x_3, x_2, x_1); \\ g_3(x_3, x_2, x_1), \end{cases}$$

требуется:

- провести минимизацию аппаратных ресурсов устройства, которое будет реализовывать заданную систему;
- построить структурную схему логического устройства в произвольном базисе по найденной минимальной форме.

Правильность построений подтвердить результатами моделирования в программе MicroCAP.

Таблица 2.10 – Таблица истинности недоопределенной логической функции

X_3	X_2	X_1	$F(X_3, X_2, X_1)$
0	0	0	---
0	0	1	0
0	1	0	1
0	1	1	---
1	0	0	1
1	0	1	---
1	1	0	---
1	1	1	1

Таблица 2.11 – Таблица истинности системы логических функций

X_3	X_2	X_1	$G_1(X_3, X_2, X_1)$	$G_2(X_3, X_2, X_1)$	$G_3(X_3, X_2, X_1)$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	1	1
0	1	1	1	0	0
1	0	0	1	1	1
1	0	1	0	0	1
1	1	0	1	1	1
1	1	1	0	1	0

I этап. Доопределение логической функции на запрещенных наборах аргументов.

Пусть логическая функция $f_0(x_3, x_2, x_1)$ описывает процесс доопределения исходной таблицы истинности (таблица 2.10) значениями *0*, тогда логическая функция $f_1(x_3, x_2, x_1)$ – процесс доопределения значениями *1*.

Логическое выражение функции $f_0(x_3, x_2, x_1)$ в СДНФ выглядит как:

$$f_0(x_3, x_2, x_1) = \overline{x}_3 \cdot x_2 \cdot \overline{x}_1 + x_3 \cdot \overline{x}_2 \cdot \overline{x}_1 + x_3 \cdot x_2 \cdot x_1 .$$

Аналогичное выражение функции $f_1(x_3, x_2, x_1)$ записывается как:

$$f_1(x_3, x_2, x_1) = \overline{x}_3 \cdot \overline{x}_2 \cdot \overline{x}_1 + \overline{x}_3 \cdot x_2 \cdot \overline{x}_1 + \overline{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot \overline{x}_2 \cdot \overline{x}_1 +$$
$$+ x_3 \cdot \overline{x}_2 \cdot x_1 + x_3 \cdot x_2 \cdot \overline{x}_1 + x_3 \cdot x_2 \cdot x_1 .$$

Логическая функция $f_1(x_3, x_2, x_1)$, представленная в СДНФ, преобразуется к сокращенной форме по методу Квайна. Напомним, что для преобразования к сокращенной форме используются три теоремы булевой алгебры:

1. $w \cdot z + w \cdot \overline{z} = w \cdot (z + \overline{z}) = w$ – теорема склеивания;
2. $w + w \cdot z = w \cdot (1 + z) = w$ – теорема поглощения;
3. $a + F \cdot \overline{a} = a + F$.

Процесс преобразования к сокращенной форме выглядит следующим образом:

$$f_1(x_3, x_2, x_1) = \overline{x}_3 \cdot \overline{x}_2 \cdot \overline{x}_1 + \overline{x}_3 \cdot x_2 \cdot \overline{x}_1 + \overline{x}_3 \cdot x_2 \cdot x_1 + x_3 \cdot \overline{x}_2 \cdot \overline{x}_1 + x_3 \cdot \overline{x}_2 \cdot x_1 +$$
$$+ x_3 \cdot x_2 \cdot \overline{x}_1 + x_3 \cdot x_2 \cdot x_1 = \overline{x}_2 \cdot \overline{x}_1 \cdot (\overline{x}_3 + x_3) + \overline{x}_3 \cdot x_2 \cdot (\overline{x}_1 + x_1) + x_3 \cdot x_2 \cdot (\overline{x}_1 + x_1) +$$
$$+ x_3 \cdot \overline{x}_2 \cdot x_1 = \overline{x}_2 \cdot \overline{x}_1 + \overline{x}_3 \cdot x_2 + x_3 \cdot x_2 + x_3 \cdot \overline{x}_2 \cdot x_1 = \overline{x}_2 \cdot \overline{x}_1 + \overline{x}_3 \cdot x_2 + x_3 \cdot (x_2 + \overline{x}_2 \cdot x_1) =$$
$$= \overline{x}_2 \cdot \overline{x}_1 + \overline{x}_3 \cdot x_2 + x_3 \cdot x_2 + x_3 \cdot x_1 = x_2 \cdot (\overline{x}_3 + x_3) + \overline{x}_2 \cdot \overline{x}_1 + x_3 \cdot x_1 =$$
$$= x_2 + \overline{x}_2 \cdot \overline{x}_1 + x_3 \cdot x_1 = x_2 + \overline{x}_1 + x_3 \cdot x_1 = x_2 + \overline{x}_1 + x_3 .$$

Составляется импликантная матрица (таблица 2.12) функции $f_0(x_3, x_2, x_1)$ и простых импликант $f_1(x_3, x_2, x_1)$. Напомним, что крестиками отмечаются те столбцы членов СДНФ, которые поглощаются отдельными простыми импликантами.

Таблица 2.12 – Импликантная матрица

ПРОСТЫЕ ИМПЛИКАНТЫ ФУНКЦИИ F_1	$\overline{x}_3 x_2 \overline{x}_1$	$x_3 \overline{x}_2 \overline{x}_1$	$x_3 x_2 x_1$
X_3		×	×
X_2	×		×
$\overline{x}_1$	×	×	

Минимальная форма логического выражения функции может быть получена тремя способами:

$$f(x_3, x_2, x_1) = x_3 + \overline{x}_1 \text{ или}$$
$$f(x_3, x_2, x_1) = x_2 + \overline{x}_1 \text{ или} \qquad (2.2)$$
$$f(x_3, x_2, x_1) = x_3 + x_2.$$

II этап. Построение структурной схемы логического устройства с одним выходом.

При выполнении этапа исследования использованы приемы №2, 3, 4, 5, 6 раздела «Типовые приемы работы в MicroCAP…».

По условию лабораторного задания структурная схема логического устройства может быть реализована в произвольном базисе. Среди вариантов минимальных форм в (2.2) наиболее выгодным с точки зрения аппаратной реализации является последнее выражение. Аппаратная реализация логической функции $f(x_3, x_2, x_1) = x_3 + x_2$ потребует применения лишь одного элемента 2ИЛИ. Нетрудно, убедиться, что перевод любого из выражений (2.2) в базисы И-НЕ, ИЛИ-НЕ для данного случая невыгоден – вновь полученные выражения будут более громоздкими.

Структурная схема логического устройства на основе минимально возможной формы $f(x_3, x_2, x_1) = x_3 + x_2$, подготовленная в программе MicroCAP, представлена на рисунке 2.5.

Особенности установки параметров для источника цифрового сигнала и прочие аспекты, связанные с моделирование схемы во временной области, подробно рассмотрены в методическом примере лабораторной работы №1.

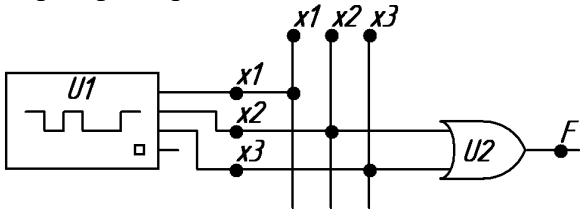


Рисунок 2.5 – Структурная схема логического устройства по минимально возможной форме

Нижняя зависимость $d(f)$ на временной диаграмме (рисунок 2.6) представляет собой значения логической функции $f(x_3, x_2, x_1)$. Сравнивая значения функции на 2, 3, 5 и 8-ом временных интервалах с соответствующими строками таблицы истинности (таблица 2.10), приходим к выводу об адекватности проведенного моделирования. На остальных временных интервалах значения функции являются факультативными (необязательными). Эти временные интервалы соответствуют запрещенным наборам аргументов таблицы истинности.

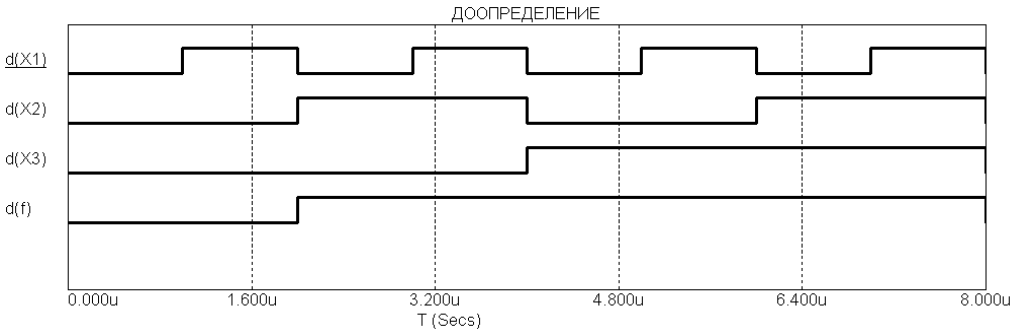


Рисунок 2.6 – Временная диаграмма работы схемы
в минимально возможной форме

III этап. Минимизация системы логических функций.

Система логических функций трех аргументов

$$\begin{cases} g_1(x_3, x_2, x_1); \\ g_2(x_3, x_2, x_1); \\ g_3(x_3, x_2, x_1), \end{cases}$$

задана в виде таблицы истинности (таблица 2.11). В первый столбец таблицы 2.13 записываются наборы аргументов, на которых хотя бы одна из выходных функций имеет значение *1*. Во второй столбец в качестве признака записываются функции, принимающие значение *1* при данном наборе аргументов.

Таблица 2.13 – Признаки наборов аргументов

Наборы аргументов	Признаки наличия единицы	Признак поглощения таблицей 2.14
$\bar{x}_3 x_2 \bar{x}_1$	$g_1 g_2 g_3$	√
$\bar{x}_3 x_2 x_1$	g_3	√
$x_3 \bar{x}_2 \bar{x}_1$	$g_1 g_2 g_3$	√
$x_3 \bar{x}_2 x_1$	g_1	√
$x_3 x_2 \bar{x}_1$	$g_1 g_2 g_3$	√
$x_3 x_2 x_1$	g_2	√

Проведем операцию склеивания над набором аргументов таблицы 2.13. Напомним, что операция склеивания возможна только над наборами аргументов, в признаках которых имеются общие функции. В операции склеивания участвуют следующие пары:

1. $\bar{x}_3 \cdot x_2 \cdot \bar{x}_1$ и $\bar{x}_3 \cdot x_2 \cdot x_1$;
2. $x_3 \cdot \bar{x}_2 \cdot \bar{x}_1$ и $x_3 \cdot \bar{x}_2 \cdot x_1$;
3. $x_3 \cdot x_2 \cdot \bar{x}_1$ и $x_3 \cdot x_2 \cdot x_1$;
4. $\bar{x}_3 \cdot x_2 \cdot \bar{x}_1$ и $x_3 \cdot x_2 \cdot \bar{x}_1$;
5. $x_3 \cdot \bar{x}_2 \cdot \bar{x}_1$ и $x_3 \cdot x_2 \cdot \bar{x}_1$.

Результаты операций склеивания заносятся в первый столбец таблицы 2.14, во втором столбце – общие признаки склеенной пары.

Таблица 2.14 – Результаты операций склеивания

ИМПЛИКАНТЫ	ПРИЗНАКИ
$\bar{x}_3 \cdot x_2$	G_3
$x_3 \cdot \bar{x}_2$	G_1
$x_3 \cdot x_2$	G_2
$x_2 \cdot \bar{x}_1$	$G_1G_2G_3$
$x_3 \cdot \bar{x}_1$	$G_1G_2G_3$

Проведем операцию поглощения членами таблицы 2.14 членов таблицы 2.13. Операция поглощения возможна только над членами, имеющими *одинаковую* комбинацию функций в признаках. Следовательно, в операции поглощения участвуют пары:

1. $\bar{x}_3 \cdot x_2$ и $\bar{x}_3 \cdot x_2 \cdot x_1$;
2. $x_3 \cdot \bar{x}_2$ и $x_3 \cdot \bar{x}_2 \cdot x_1$;
3. $x_3 \cdot x_2$ и $x_3 \cdot x_2 \cdot x_1$;
4. $x_2 \cdot \bar{x}_1$ и $\bar{x}_3 \cdot x_2 \cdot \bar{x}_1$;
5. $x_3 \cdot \bar{x}_1$ и $x_3 \cdot x_2 \cdot \bar{x}_1$;
6. $x_2 \cdot \bar{x}_1$ и $x_3 \cdot x_2 \cdot \bar{x}_1$ или $x_3 \cdot \bar{x}_1$ и $x_3 \cdot x_2 \cdot \bar{x}_1$.

По результатам проведения операции следует, что все члены таблицы 2.13 поглотились. По этой причине в импликантной матрице (таблица 2.15) в первом столбце перечислены только члены из таблицы 2.14. В противном случае, следовало бы дописать непоглощенные члены таблицы 2.13 в первый столбец таблицы 2.15 (см. пример раздела 2.3). В остальных столбцах матрицы записаны члены таблицы 2.13 до их поглощения.

Таблица 2.15 – Импликантная матрица

Импликанта	$\bar{x}_3x_2\bar{x}_1$			$\bar{x}_3x_2x_1$	$x_3\bar{x}_2\bar{x}_1$			$x_3\bar{x}_2x_1$	$x_3x_2\bar{x}_1$			$x_3x_2x_1$
	g_1	g_2	g_3	g_3	g_1	g_2	g_3	g_1	g_1	g_2	g_3	g_2
$\bar{x}_3x_2\,(g_3)$			×	(×)								
$x_3\bar{x}_2\,(g_1)$					×			(×)				
$x_3x_2\,(g_2)$										×		(×)
$x_2\bar{x}_1\,(g_1g_2g_3)$	(×)	×	×						(×)	×	×	
$x_3\bar{x}_1\,(g_1g_2g_3)$					(×)	×	×		×	×	×	

В импликантной матрице крестиками отмечены столбцы членов, поглощаемых отдельными импликантами с учетом общих функциональных признаков.

Определим набор импликант, обеспечивающий перекрытие всех столбцов матрицы (таблица 2.16).

Таблица 2.16 – Набор импликант, обеспечивающий перекрытие

ИМПЛИКАНТЫ	ПРИЗНАКИ
$x_2 \cdot \bar{x}_1$	$G_1G_2G_3$
$\bar{x}_3 \cdot x_2$	G_3
$x_3 \cdot \bar{x}_1$	$G_1G_2G_3$
$x_3 \cdot \bar{x}_2$	G_1
$x_3 \cdot x_2$	G_2

Заметим, что в данном примере перекрытие столбца $x_3 \cdot x_2 \cdot \bar{x}_1$ возможно двумя равноценными способами: импликантой $x_2 \cdot \bar{x}_1$ (как это сделано в таблице 2.15) или импликантой $x_3 \cdot \bar{x}_1$.

Запишем для выходных функций логические выражения, составленные из импликант таблицы 2.16, в признаках которых содержатся заданные функции:

$$\begin{aligned} g_1(x_3, x_2, x_1) &= x_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_2; \\ g_2(x_3, x_2, x_1) &= x_2 \cdot \bar{x}_1 + x_3 \cdot \bar{x}_1 + x_3 \cdot x_2; \\ g_3(x_3, x_2, x_1) &= x_2 \cdot \bar{x}_1 + \bar{x}_3 \cdot x_2 + x_3 \cdot \bar{x}_1. \end{aligned} \qquad (2.3)$$

Анализ полученных выражений позволяет сделать вывод: импликанты $x_2 \cdot \bar{x}_1$ и $x_3 \cdot \bar{x}_1$ присутствуют одновременно во всех логических функциях. Это позволяет значительно сократить (минимизировать) общее количество логических элементов при синтезе устройства.

IV этап. Построение структурной схемы логического устройства с тремя выходами.

При выполнении этапа исследования использованы приемы №2, 3, 4, 5, 6 раздела «Типовые приемы работы в MicroCAP…».

По условию лабораторного задания структурная схема логического устройства может быть реализована в произвольном базисе. Аппаратная реализация системы логических функций (2.3) потребует 11 логических элементов. В частности, три элемента НЕ нужны для выработки инверсных сигналов $\bar{x}_3$, $\bar{x}_2$, $\bar{x}_1$; пять элементов 2И требуются для соответствующих операций логического умножения; три элемента 3ИЛИ будут формировать окончательный вид логических функций – дизъюнкцию конъюнктивных членов.

Структурная схема логического устройства на основе системы (2.3), подготовленная в программе MicroCAP, представлена на рисунке 2.7. Все технические подробности, связанные с изображением схемы в MicroCAP и с последующим моделированием во временной области, были рассмотрены ранее.

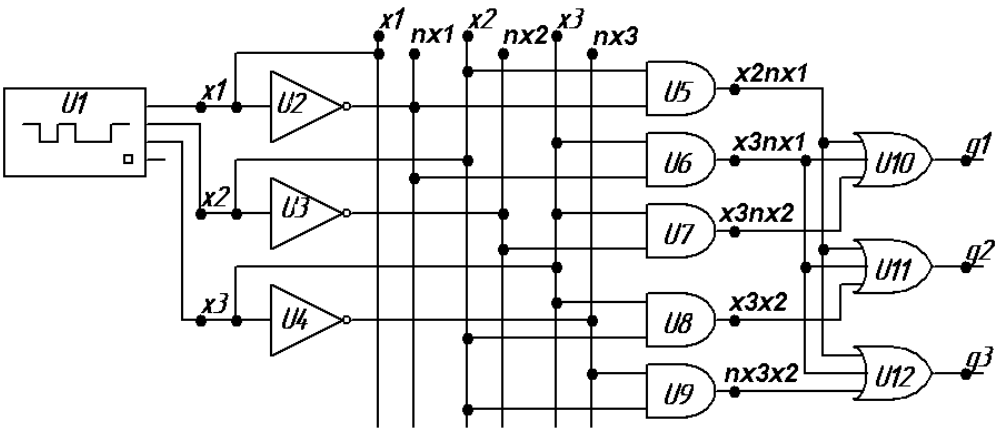


Рисунок 2.7 – Структурная схема логического устройства
с тремя выходами

Временная диаграмма работы устройства с тремя выходами показана на рисунке 2.8. На графике для удобства восприятия и анализа, помимо одиночных сигналов, присутствуют временные зависимости входной $x_3x_2x_1$ и выходной $g_3g_2g_1$ шин данных.

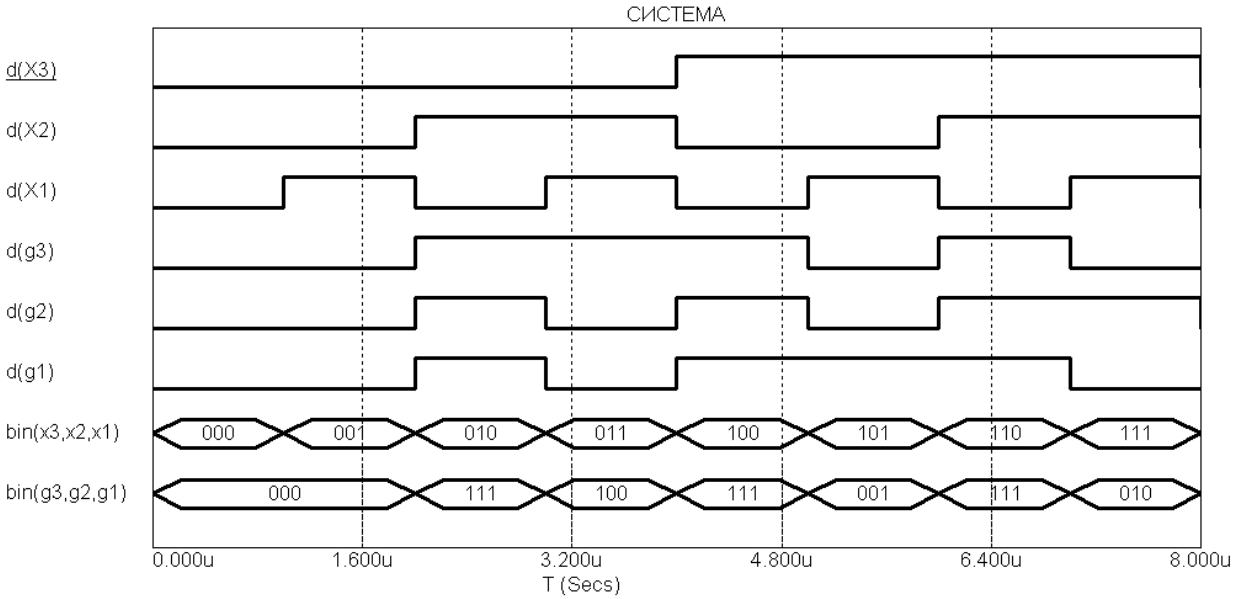


Рисунок 2.8 – Временная диаграмма работы
логического устройства с тремя выходами

2.5 Лабораторное задание

Повторить методический пример, приведенный выше, по исходным данным Вашего варианта. *Критерием правильности* проведенного исследования является совпадение значений выходного сигнала на различных временных интервалах в MicroCAP с соответствующими строками таблицы истинности.

2.6 Контрольные вопросы

1. В чем заключаются основные этапы минимизации логических функций по методу Квайна?
2. Что такое операция склеивания?
3. Что такое операция поглощения?
4. Чем отличается сокращенная форма представления логической функции от минимальной?
5. Что такое простая импликанта?
6. Каков принцип заполнения импликантной матрицы?
7. Что такое ядро сокращенной формы представления логической функции?
8. Что нужно сделать, чтобы на основе заполненной импликантной матрицы получить минимальную форму логической функции?
9. Что такое недоопределенная логическая функция?
10. В чем особенность минимизации системы логических функций?

2.7 Варианты заданий

2.7.1 Варианты заданий для минимизации недоопределенных логических функций

Для всех вариантов задания общими являются три столбца таблицы истинности (таблица 2.17), которые представляют собой восемь возможных сочетаний аргументов логической функции $f(x_3, x_2, x_1)$. Варианты значений логической функции приведены в таблице 2.18. Выбрав соответствующий своему варианту столбец в таблице 2.18, необходимо присоединить его в качестве четвертого столбца к таблице 2.17 и получить, таким образом, полноценную таблицу истинности.

Таблица 2.17 – Сочетания аргументов логической функции

X_3	X_2	X_1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Таблица 2.18 – Варианты значений логической функции $f(x_3, x_2, x_1)$

	Номера вариантов												
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*
Значения функции	–	–	1	–	1	–	1	1	–	–	0	0	0
	–	1	–	1	–	0	0	–	–	0	1	–	–
	–	–	–	1	–	–	1	1	–	–	1	–	1
	–	0	–	1	0	–	0	–	1	1	–	1	–
	1	–	–	0	1	–	–	0	–	–	–	–	–
	0	0	0	–	1	0	–	–	0	0	–	0	1
	1	–	1	–	–	1	–	–	1	1	1	1	1
	0	0	0	–	–	1	–	0	0	–	–	–	–

	Номера вариантов											
	14	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*
Значения функции	0	1	–	0	–	0	1	1	1	–	0	–
	–	0	0	0	–	–	0	1	1	0	–	0
	0	–	–	–	0	–	1	0	1	–	–	–
	0	0	–	–	1	1	–	–	–	–	–	–
	–	–	1	1	–	–	–	–	–	–	–	–
	–	1	1	–	1	–	–	–	–	1	0	1
	–	–	–	0	–	0	0	0	0	0	1	1
	1	–	1	–	0	1	–	–	–	1	1	1

2.7.2 Варианты заданий для минимизации системы логических функций

Для всех вариантов задания общими являются три столбца таблицы истинности (таблица 2.19), которые представляют собой восемь возможных сочетаний аргументов для логических функций $g_1(x_3, x_2, x_1)$, $g_2(x_3, x_2, x_1)$, $g_3(x_3, x_2, x_1)$. Варианты значений логических функций приведены в таблице 2.20. Выбрав соответствующие своему варианту три столбца в таблице 2.20, необходимо присоединить их к таблице 2.19 и получить, таким образом, полноценную таблицу истинности.

Таблица 2.19 – Сочетания аргументов логических функций

X_3	X_2	X_1
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Таблица 2.20 – Варианты значений логических функций $g_1(x_3, x_2, x_1)$, $g_2(x_3, x_2, x_1)$, $g_3(x_3, x_2, x_1)$

	Номера вариантов																	
	1			*2*			*3*			*4*			*5*			*6*		
Функции	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3
Значения функций	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0
	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0
	1	1	0	1	0	0	0	0	1	0	1	0	1	0	1	0	1	0
	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	1
	1	1	1	1	1	0	1	0	0	0	0	1	0	1	0	1	0	1
	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0
	Номера вариантов																	
	7			*8*			*9*			*10*			*11*			*12*		
Функции	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3
Значения функций	1	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	1	0	1
	1	1	1	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0
	0	0	1	0	1	1	1	1	0	1	0	1	0	1	0	1	0	0
	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	0	0
	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1
	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1
	Номера вариантов																	
	13			*14*			*15*			*16*			*17*			*18*		
Функции	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3
Значения функций	0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	1
	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	0
	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0	1	0	0
	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0
	1	1	1	1	1	1	1	1	0	1	0	1	0	1	1	1	1	1
	1	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0
	Номера вариантов																	
	19			*20*			*21*			*22*			*23*			*24*		
Функции	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3
Значения функций	0	1	1	0	1	0	1	0	0	0	0	1	0	1	0	1	0	0
	1	1	1	1	1	0	1	1	0	0	1	1	0	1	1	1	0	1
	1	0	0	0	0	1	0	0	1	1	0	0	1	0	0	0	1	0
	0	0	0	1	0	1	0	1	1	1	1	0	0	1	1	1	0	1
	0	1	0	0	1	0	1	0	0	0	0	1	1	0	0	0	1	0
	1	1	1	1	0	1	0	1	1	1	1	0	1	0	1	0	1	1
	0	0	1	0	1	0	1	0	0	0	0	1	0	1	0	1	0	0
	1	1	0	0	1	1	1	0	1	1	0	1	0	1	1	1	0	1

Окончание таблицы 2.20

	Номер варианта		
	25		
Функции	G_1	G_2	G_3
Значения функций	0	1	0
	1	0	0
	0	1	1
	1	0	0
	0	0	1
	1	1	1
	0	1	0
	1	1	0

3 ЛАБОРАТОРНАЯ РАБОТА №3 – УНИВЕРСАЛЬНЫЕ ЛОГИЧЕСКИЕ МОДУЛИ НА ОСНОВЕ МУЛЬТИПЛЕКСОРОВ

3.1 Цель работы

В ходе выполнения настоящей работы предусматривается:

1) знакомство с принципом действия мультиплексора, его режимами работы, назначением выводов;

2) изучение некоторых способов настройки универсальных логических модулей на основе мультиплексоров;

3) приобретение навыков разложения логических выражений по Шеннону для случая двух переменных;

4) отработка приемов по составлению остаточных функций для универсальных логических модулей.

3.2 Порядок выполнения работы

1. Изучить методические указания к лабораторной работе.
2. Письменно, в отчете по лабораторной работе ответить на контрольные вопросы.
3. Внимательно ознакомиться с примером, приведенным в пункте 3.4.
4. Выполнить лабораторное задание согласно варианту задания.
5. Сделать выводы по работе.

Внимание! Отчет по лабораторной работе в обязательном порядке должен содержать: схемы включения, графики зависимостей, все необходимые расчеты и их результаты, текстовые пояснения. На графиках в отчете должны присутствовать единицы измерения, масштаб, цена деления.

3.3 Способы настройки универсальных логических модулей

Универсальные логические модули (УЛМ) на основе мультиплексоров относятся к устройствам, настраиваемым на решение той или иной задачи. Универсальность их состоит в том, что для заданного числа аргументов можно настроить УЛМ на любую функцию. Известно, что общее число функций n аргументов выражается как 2^{2^n}. С ростом n число функций растет чрезвычайно быстро. Хотя практический интерес представляет не все существующие функции, возможность получить любую из огромного числа функций свидетельствует о больших перспективах применения УЛМ.

Первый способ настройки УЛМ. Первым способом настройки, используемым в УЛМ, является фиксация некоторых входов. Для этого способа справедливо следующее соотношение между числом аргументов и числом настроечных входов. Пусть число аргументов n и требуется настройка на любую из функций. Тогда число комбинаций для кода настройки, равное числу функций, есть 2^{2^n}. Для двоичного кода число комбинаций связано с разрядностью кода выражением 2^m, где m – разрядность кода. Приравнивая число воспроизводимых функций к числу комбинаций кода настройки, имеем для числа настроечных входов соотношение $m = 2^n$.

Полученному выражению отвечает соотношение между числом входов разного типа для мультиплексора. При этом на адресные входы следует подавать аргументы функции, а на

информационные входы – сигналы настройки (рисунок 3.1). Таким образом, для использования мультиплексора в качестве УЛМ следует изменить назначение его входов.

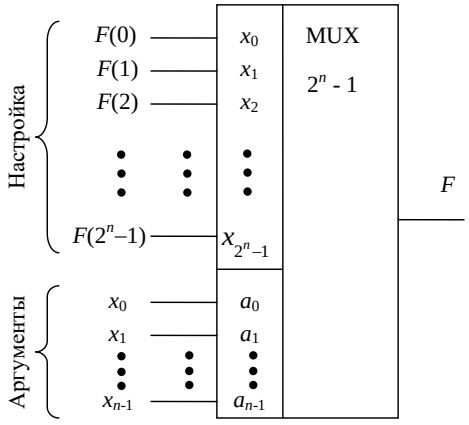


Рисунок 3.1 – Схема использования мультиплексора в качестве УЛМ

Рисунок 3.1 иллюстрирует возможность воспроизведения с помощью мультиплексора любой функции n аргументов. Действительно, каждому набору аргументов соответствует передача на выход одного из сигналов настройки. Если этот сигнал есть значение функции на данном наборе аргументов, то задача решена. Разным функциям будут соответствовать разные коды настройки. Алфавитом настройки будет {0, 1} – настройка осуществляется константами 0 и 1. На рисунке 3.2 показан пример воспроизведения функции неравнозначности $x_1 \oplus x_2$ с помощью мультиплексора «4 – 1».

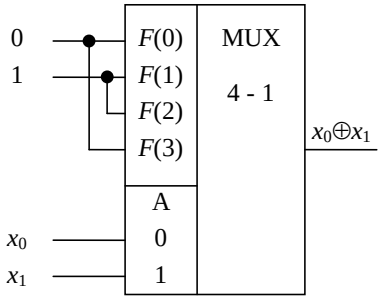


Рисунок 3.2 – Воспроизведение функции при настройке константами

Большое число настроечных входов затрудняет реализацию УЛМ. Для УЛМ, расположенных внутри кристалла, можно вводить код настройки последовательно в сдвигающий регистр, к разрядам которого подключены входы настройки. Тогда внешним входом настройки будет всего один, но настройка будет занимать не один такт, а 2^n тактов. Возможны и промежуточные последовательно-параллельные варианты ввода кода настройки.

Второй способ настройки УЛМ. Большое число входов настройки наталкивает на поиск возможностей их уменьшения. Такие возможности существуют и заключаются в расширении алфавита настроечных сигналов. Если от алфавита {0, 1} перейти к алфавиту {0, 1, $\tilde{x}_i$}, где $\tilde{x}_i$ - литерал одного из аргументов, то число входов аргументов сократится на единицу, а число настроечных входов – вдвое. Напомним, что под литералом переменной понимается либо сама переменная, либо ее инверсия. Перенос одного из аргументов в число сигналов настройки не влечет за собою каких-либо схемных изменений. На том же оборудовании будут реализованы функции с числом аргументов на единицу больше, чем при настройке константами.

Для нового алфавита код настройки находится следующим образом. Аргументы, за исключением $\tilde{x}_i$, подаются на адресующие входы, что соответствует их фиксации в выражении для искомой функции, которая становится функцией единственного аргумента $\tilde{x}_i$. Эту функцию, которую назовем остаточной, и нужно подавать на настроечные входы.

Если искомая функция зависит от *n* аргументов и в число сигналов настройки будет перенесен один из аргументов, то возникает *n* вариантов решения задачи, т.к. в сигналы настройки может быть перенесен любой аргумент. Спрашивается, какой именно аргумент целесообразно переносить в сигналы настройки? *Здесь можно опираться на рекомендацию: в настроечные сигналы следует переводить аргумент, который имеет минимальное число вхождений в импликанты функции.* В этом случае будут максимально использованы как бы внутренние логические ресурсы мультиплексора, а среди сигналов настройки увеличится число констант, что и считается благоприятным для схемной реализации УЛМ.

Проиллюстрируем сказанное примером воспроизведением функции трех аргументов $F = x_3x_2x_1 + \bar{x}_3\bar{x}_2$. Минимальное число вхождений в выражение функции имеет переменная x_1, которую и перенесем в число сигналов настройки. Остаточная функция определится таблицей 3.1.

Например, при $x_3 = 0$ и $x_2 = 0$ имеем $F_{OCT} = 0 \cdot 0 \cdot x_1 + \bar{0} \cdot \bar{0} = 1$. При $x_3 = 0$ и $x_2 = 1$ имеем $F_{OCT} = 0 \cdot 1 \cdot x_1 + \bar{0} \cdot \bar{1} = 0$ и т.д.

Схема УЛМ приведена на рисунке 3.3.

По пути расширения алфавита сигналов настройки можно идти и дальше, но при этом понадобятся дополнительные логические схемы, воспроизводящие остаточные функции, которые будут уже зависеть более чем от одного аргумента.

Таблица 3.1 – Таблица истинности остаточной функции

X_3	X_2	$F_{ОСТ}$
0	0	1
0	1	0
1	0	0
1	1	X_1

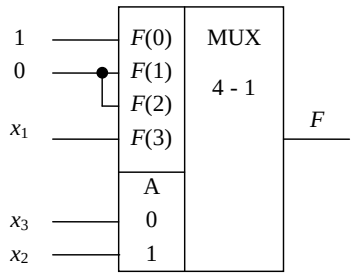


Рисунок 3.3 – Перенос одного аргумента в число сигналов настройки

Если в сигналы настройки перевести два аргумента, то дополнительные логические схемы будут двухвходовыми вентилями, что мало усложняет УЛМ и может оказаться приемлемым решением. В этом случае для сохранения универсальности УЛМ мультиплексору нужно предпослать блок выработки остаточных функций, в котором формируются все функции двух переменных (за исключением констант 0, 1 и литералов самих переменных, которые не требуется вырабатывать). Такой блок показан на рисунке 3.4. Пример реализации функции $F = x_2x_1 + \bar{x}_4x_3$ при алфавите настройки {0, 1, $\tilde{x}_1$, $\tilde{x}_2$} показан на рисунке 3.5. Таблица остаточной функции для этого примера приведена в таблице 3.2.

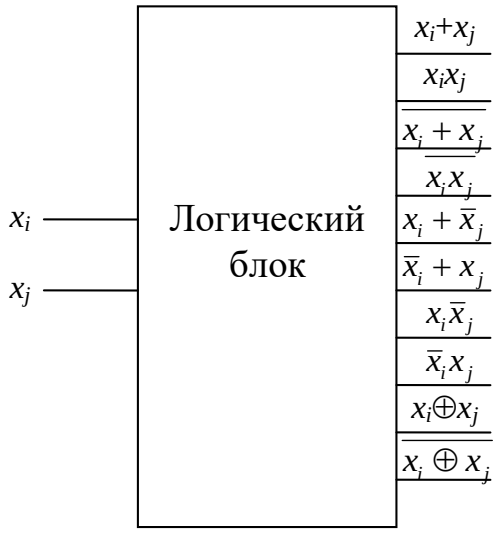


Рисунок 3.4 – Логический блок выработки сигналов настройки УЛМ с переносом двух аргументов в сигналы настройки

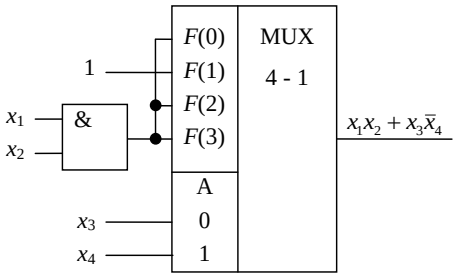


Рисунок 3.5 – Схема воспроизведения функции четырех аргументов на мультиплексоре «4 – 1»

Таблица 3.2 – Таблица истинности остаточной функции

X_4	X_3	$F_{ОСТ}$
0	0	X_2X_1
0	1	1
1	0	X_2X_1
1	1	X_2X_1

Пирамидальные структуры УЛМ. Дальнейшее расширение алфавита настройки за счет переноса трех и более переменных в сигналы настройки требует вычислений остаточных функций трех или более переменных. Вычисление таких остаточных функций с помощью мультиплексоров приводит к пирамидальной структуре (рисунок 3.6), в которой мультиплексоры первого яруса реализуют остаточные функции, а мультиплексор второго яруса вырабатывает искомую функцию.

Показанная пирамидальная структура – каноническое решение, которое приводит к нужному результату, но не претендует на оптимальность. Дело в том, что варианты построения схем из нескольких мультиплексоров для воспроизведения функций многих переменных разнообразны, но алгоритм поиска оптимального по затратам оборудования или какому-либо другому критерию отсутствует. Имеются работы, в которых найдены решения более высокого качества, но это результаты изобретений, касающиеся частных случаев, и не относятся к регулярному методу поиска структур.

При чисто электронной настройке константами 0 и 1 схема воспроизводит функцию n аргументов, где $n = k + p$, причем k – число аргументов, подаваемых на мультиплексор вто-

рого яруса, p – число аргументов, от которых зависят остаточных функции, воспроизводимые мультиплексорами $0\ldots(2^k - 1)$ первого яруса.

Для уменьшения аппаратных затрат в схеме следует стремиться к минимизации числа мультиплексоров в столбце, т.е. минимизации k и, соответственно, максимальным p, поскольку их сумма k + p постоянна и равна n.

Сигналы настройки для мультиплексоров первого яруса можно искать разными способами:

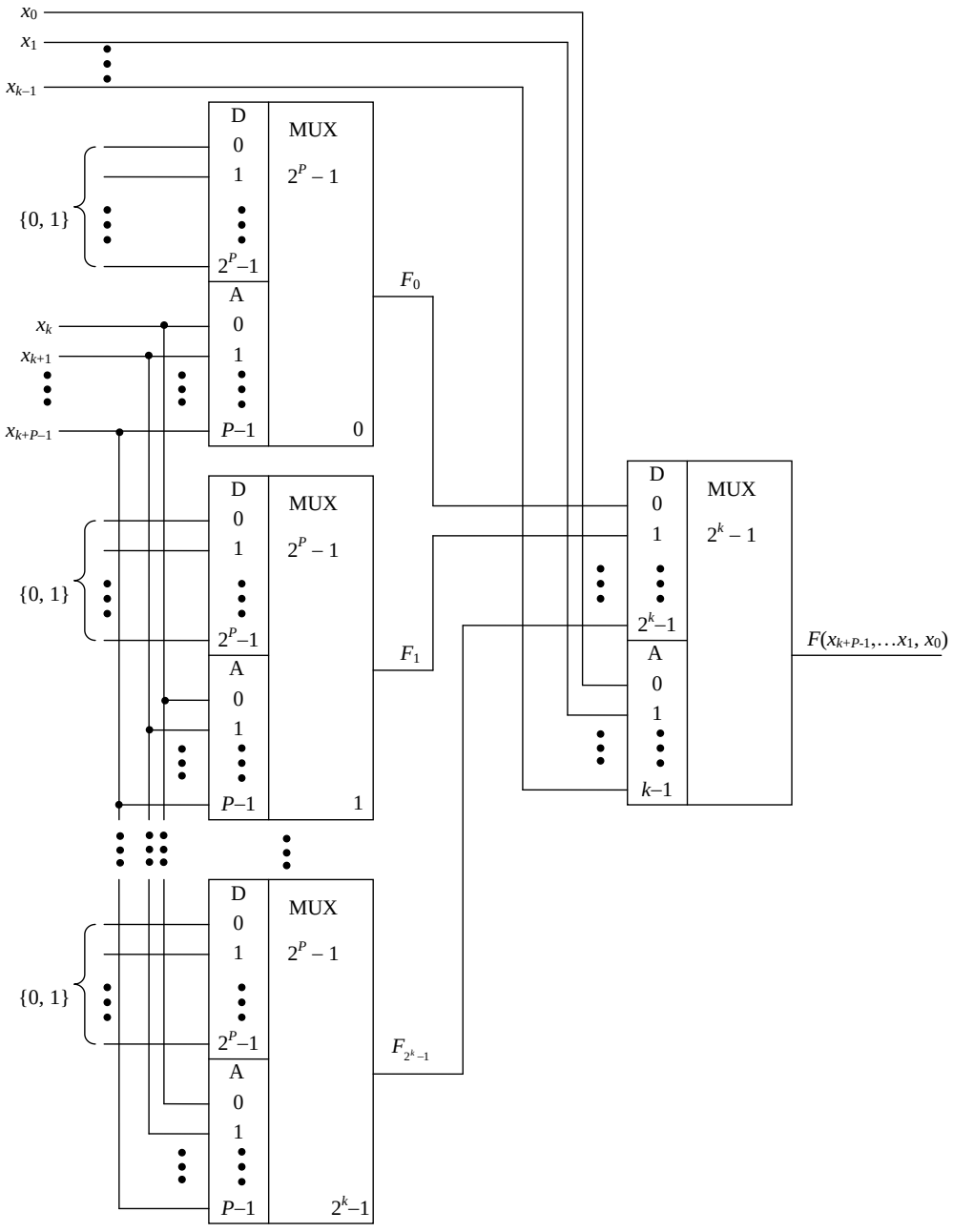


Рисунок 3.6 – Структура УЛМ,
построенного на нескольких мультиплексорах

1. Подстановкой (фиксацией) наборов аргументов, подаваемых на адресные входы мультиплексоров для получения остаточных функций и, далее, сигналов настройки. Этот способ уже рассмотрен (таблицы 3.1, 3.2).

2. С помощью разложения функции по Шеннону. Это разложение можно произвести по разному числу переменных. По одному из аргументов разложение имеет вид:

$$F\left(x_{n-1},...,x_1,x_0\right)=\overline{x}_0\cdot F\left(x_{n-1},...,x_1,0\right)+x_0\cdot F\left(x_{n-1},...,x_1,1\right).$$

Справедливость такого разложения видна из подстановки в него значений $x_0 = 0$ и $x_0 = 1$, что дает непосредственно функции $F(x_{n-1}, \ldots, x_1, 0)$ и $F(x_{n-1}, \ldots, x_1, 1)$.

Разложение функции по двум аргументам:

$$F\left(x_{n-1},...,x_1,x_0\right)=\overline{x}_1\overline{x}_0\cdot F\left(x_{n-1},...,x_2,0,0\right)+\overline{x}_1x_0\cdot F\left(x_{n-1},...,x_2,0,1\right)+$$

$$+x_1\overline{x}_0\cdot F\left(x_{n-1},...,x_2,1,0\right)+x_0x_1\cdot F\left(x_{n-1},...,x_2,1,1\right)$$

и, наконец, разложение по k аргументам:

$$F\left(x_{n-1},...,x_1,x_0\right)=\overline{x}_{k-1}\overline{x}_{k-2}...\overline{x}_1\overline{x}_0\cdot F\left(x_{n-1},...,x_k,0,...,0,0\right)+$$

$$+\overline{x}_{k-1}\overline{x}_{k-2}...\overline{x}_1x_0\cdot F\left(x_{n-1},...,x_k,0,...,0,1\right)+...$$

$$...+x_{k-1}x_{k-2}...x_1x_0\cdot F\left(x_{n-1},...,x_k,1,...,1,1\right)=$$

$$=\overline{x}_{k-1}\overline{x}_{k-2}...\overline{x}_1\overline{x}_0\cdot F_0+\overline{x}_{k-1}\overline{x}_{k-2}...\overline{x}_1x_0\cdot F_1+...+x_{k-1}x_{k-2}...x_1x_0\cdot F_{2^k-1},$$

где

$$F_0 = F(x_{n-1}, \ldots, x_k, 0, \ldots, 0, 0),$$
$$F_1 = F(x_{n-1}, \ldots, x_k, 0, \ldots, 0, 1),$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$F_{2^k-1}=F\left(x_{n-1},...,x_k,1,...,1,1\right).$$

Структура формул разложения полностью соответствует реализации двухъярусных УЛМ. В первом ярусе реализуются функции F_i, ($i = 0, \ldots, 2^k–1$), зависящие от $n – k$ аргументов, которые используются как настроечные для второго яруса, мультиплексор которого воспроизводит функцию k аргументов.

3. Сигналы настройки можно получить непосредственно из таблицы истинности функции. *Для удобства просмотра таблицы ее следует записать так, чтобы аргументы, переносимые в сигналы настройки, играли роль младших разрядов в словах-наборах аргументов.* Пусть имеется функция четырех переменных $f(x_3, x_2, x_1, x_0)$, и переменная x_3 считается старшим разрядом вектора аргументов. Пусть, далее, функция задана перечислением наборов аргументов, на которых она принимает единичные значения, причем заданы десятичные значения этих наборов: 3, 4, 5, 6, 7, 11, 15. Заметим, что аналитическое значение этой функции имеет вид $F = x_0x_1 + x_2\overline{x}_3$. Значения функции сведены в таблицу 3.3.

При электронной настройке УЛМ константами 0 и 1 требуется мультиплексор размерности «16 – 1», на настроечные входы УЛМ подаются значения самой функции из таблицы.

При переносе литерала $\tilde{x}_0$ в сигналы настройки (алфавит настройки $\{0, 1, \tilde{x}_0\}$) требуется найти остаточную функцию, аргументами которой является вектор переменных x_3, x_2, x_1. Каждая комбинация этих переменных встречается в двух смежных строках таблицы. Просматривая таблицу по смежным парам строк, можно видеть, что остаточная функция соответствует другой таблице (таблица 3.4).

Для реализации этого варианта УЛМ достаточен мультиплексор «8 – 1», но для перестройки на другую функцию потребуется не только смена кода настройки, но и коммутация входов настройки для подачи литералов переменной на другие настроечные входы.

При переносе в сигналы настройки двух переменных ($\tilde{x}_0$ и $\tilde{x}_1$) для поиска остаточных функция следует просмотреть четверки смежных строк таблицы с неизменными наборами x_3x_2 – аргументами, подаваемыми на адресные входы УЛМ. Этот просмотр приводит к следующей таблице истинности (таблица 3.5).

Из таблицы видно, что для воспроизведения функции достаточно использовать мультиплексор «4 – 1» с дополнительным конъюнктором для получения произведения x_1x_0. Но при перестройке на другую функцию потребуются и другие функции двух переменных, т.е. универсальный логический модуль должен включать в свой состав дополнительный логический блок (см. рисунок 3.4).

Таблица 3.3 – Таблица истинности функции четырех аргументов

X_3	X_2	X_1	X_0	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Таблица 3.4 – Таблица истинности остаточной функции трех аргументов

X_3	X_2	X_1	F
0	0	0	0
0	0	1	X_0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	X_0
1	1	0	0
1	1	1	X_0

Таблица 3.5 – Таблица истинности остаточной функции двух аргументов

X_3	X_2	F
0	0	X_1X_0
0	1	1
1	0	X_1X_0
1	1	X_1X_0

3.4 Пример различных способов настройки УЛМ

Логические функции $f(x_2, x_1, x_0)$, $g(x_3, x_2, x_1, x_0)$ и $h(x_4, x_3, x_2, x_1, x_0)$ заданы в виде следующих булевых алгебраических выражений:

$$f(x_2, x_1, x_0) = x_0 + \bar{x}_2 x_1; \tag{3.1}$$

$$g(x_3, x_2, x_1, x_0) = x_2 x_1 x_0 + \bar{x}_3; \tag{3.2}$$

$$h(x_4, x_3, x_2, x_1, x_0) = x_2 x_0 + x_4 \bar{x}_3 x_1. \tag{3.3}$$

Для логической функции $f(x_2, x_1, x_0)$ требуется аппаратно реализовать ее на основе УЛМ первым способом – фиксацией информационных входов сигналами настройки с алфавитом {0, 1}.

Для логической функции $g(x_3, x_2, x_1, x_0)$ требуется аппаратно реализовать ее на основе УЛМ вторым способом – переносом двух аргументов в сигналы настройки и алфавитом {0, 1, $\tilde{x}_1$, $\tilde{x}_0$}.

Для логической функции $h(x_4, x_3, x_2, x_1, x_0)$ требуется аппаратно реализовать ее третьим способом – на основе пирамидальной двухъярусной структуры УЛМ.

Правильность аппаратной реализации каждым способом проверить в программе MicroCAP путем сравнения с проверочными схемами. Под проверочными схемами будем понимать структурные схемы в базисе НЕ, И, ИЛИ, выполненные непосредственно по выражениям (3.1) – (3.3).

I этап. Синтез логического устройства на основе УЛМ с алфавитом настройки {0, 1}.

При выполнении этапа исследования использованы приемы №1 – 7 раздела «Типовые приемы работы в MicroCAP…».

Логическое устройство должно быть реализовано на основе УЛМ, поэтому целесообразно, чтобы окончательным видом данной функции $f(x_2, x_1, x_0)$ являлась СДНФ. СДНФ содержит все адреса, по которым нужно хранить единичные значения функции. Набор аргументов СДНФ есть адрес конкретного информационного входа УЛМ.

Для перевода функции $f(x_2, x_1, x_0)$ в СДНФ следует конъюнктивные члены, не содержащие переменной x_i, умножить на равную единице дизъюнкцию $\left(x_i + \bar{x}_i\right)$:

$$f\left(x_2, x_1, x_0\right) = x_0 + \bar{x}_2 x_1 = x_0\left(x_2 + \bar{x}_2\right)\left(x_1 + \bar{x}_1\right) + \bar{x}_2 x_1\left(x_0 + \bar{x}_0\right) =$$

$$= \left(x_2 x_0 + \bar{x}_2 x_0\right)\left(x_1 + \bar{x}_1\right) + \bar{x}_2 x_1 x_0 + \bar{x}_2 x_1 \bar{x}_0 =$$

$$= x_2 x_1 x_0 + \bar{x}_2 x_1 x_0 + x_2 \bar{x}_1 x_0 + \bar{x}_2 \bar{x}_1 x_0 + \bar{x}_2 x_1 x_0 + \bar{x}_2 x_1 \bar{x}_0 =$$

$$= x_2 x_1 x_0 + \bar{x}_2 x_1 x_0 + x_2 \bar{x}_1 x_0 + \bar{x}_2 \bar{x}_1 x_0 + \bar{x}_2 x_1 \bar{x}_0.$$

Используя полученную СДНФ функции, восстановим таблицу истинности (таблица 3.6). Анализ таблицы 3.6 позволяет сформулировать словесное правило настройки УЛМ константами: на информационные входы D1, D2, D3, D5, D7 мультиплексора следует подать логическую единицу, на остальные информационные входы – логический ноль. При этом адресные входы мультиплексора подключаются к цифровому источнику сигналов x_2, x_1, x_0.

Таблица 3.6 – Таблица истинности функции $f(x_2, x_1, x_0)$

X_2	X_1	X_0	$F(X_2, X_1, X_0)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Структурная схема логического устройства на основе УЛМ с алфавитом настройки {0, 1}, подготовленная в программе MicroCAP, представлена в верхней части рисунка 3.7. В нижней части приведена проверочная схема, реализованная непосредственно по (3.1).

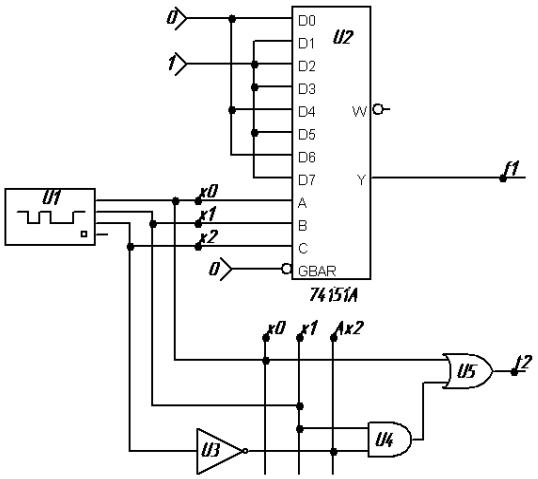
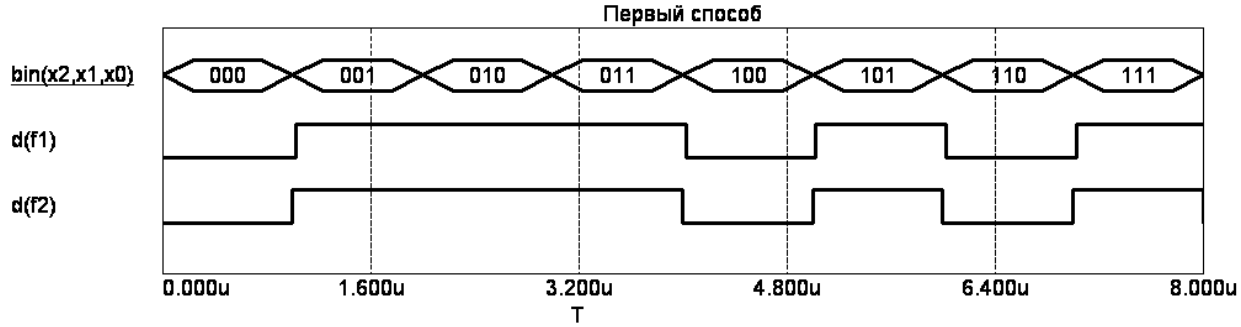


Рисунок 3.7 – Структурная и проверочная схемы логического устройства по первому способу настройки

В схеме применен селектор-мультиплексор на 8 каналов 74151А (отечественный аналог КМ155КП7). Временная зависимость входных сигналов x_2, x_1, x_0 на рисунке 3.8 представлена в виде состояния соответствующей трехразрядной шины.

Результаты моделирования говорят о совпадении в двух схемах временной зависимости выходного сигнала устройства и соответствии этого сигнала таблице истинности (таблица 3.6).



d(*f*1) – на основе УЛМ; *d*(*f*2) – на основе проверочной схемы

Рисунок 3.8 – Временная диаграмма функции $f(x_2, x_1, x_0)$

<u>II этап. Синтез логического устройства на основе УЛМ с алфавитом настройки</u> {0, 1, $\tilde{x}_1$, $\tilde{x}_0$}.

При выполнении этапа исследования использованы приемы №1 – 7 раздела «Типовые приемы работы в MicroCAP…».

Если в сигналы настройки УЛМ перевести два аргумента исходной логической функции, то число входов аргументов (адресные входы мультиплексора) сократится на два, а число настроечных входов – в четыре раза. При этом дополнительные логические схемы будут двухвходовыми вентилями, что мало усложняет УЛМ и оказывается приемлемым решением.

Поскольку в исходной логической функции $g(x_3, x_2, x_1, x_0) = x_2x_1x_0 + \bar{x}_3$ все аргументы встречаются по одному разу, можно произвольно выбрать аргументы, подлежащие переносу в сигналы настройки, например, x_1 и x_0. Для двух оставшихся аргументов x_3 и x_2 проведем верификацию четырех возможных сочетаний.

При $x_3 = 0$ и $x_2 = 0$ имеем $g(x_3, x_2) = 0 \cdot x_1x_0 + \bar{0} = 1.$

При $x_3 = 0$ и $x_2 = 1$ имеем $g(x_3, x_2) = 1 \cdot x_1x_0 + \bar{0} = 1.$

При $x_3 = 1$ и $x_2 = 0$ имеем $g(x_3, x_2) = 0 \cdot x_1 x_0 + \overline{1} = 0$.

При $x_3 = 1$ и $x_2 = 1$ имеем $g(x_3, x_2) = 1 \cdot x_1 x_0 + \overline{1} = x_1 x_0$.

По результатам приведенной верификации составим таблицу истинности остаточной функции (таблица 3.7). Из таблицы 3.7 видно, что для аппаратной реализации исходной функции $g(x_3, x_2, x_1, x_0)$ потребуется мультиплексор «4 – 1», на адресные входы которого поступают сигналы x_3, x_2; причем младший разряд x_2 должен поступать обязательно на младший адресный вход. На информационные входы D0…D2 будут поданы логические константы *1, 1, 0*, соответственно. К информационному входу D3 подключается логический элемент 2И, на входы которого поступают сигналы x_1, x_0.

Таблица 3.7 – Таблица истинности остаточной функции

X_3	X_2	$G(X_3, X_2)$
0	0	1
0	1	1
1	0	0
1	1	$X_1 \cdot X_0$

По словесному описанию можно разработать в программе MicroCAP структурную схему логического устройства на основе УЛМ (рисунок 3.9, верхний фрагмент). На нижнем фрагменте рисунка приведена проверочная схема, реализованная непосредственно по выражению (3.2).

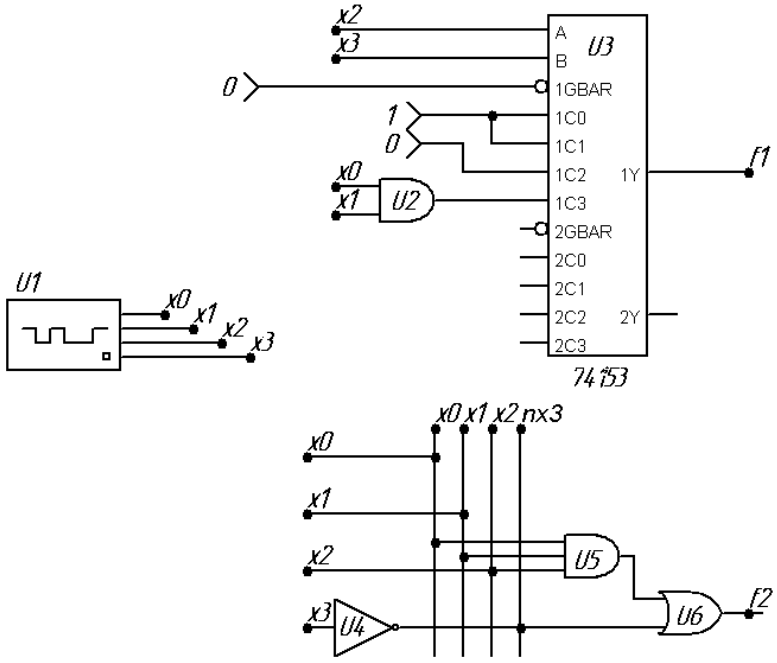


Рисунок 3.9 - Структурная и проверочная схемы логического устройства по второму способу настройки

В схеме применен сдвоенный селектор-мультиплексор 74153 типа «2 × (4 – 1)» (отечественный аналог К155КП2). Вторая независимая часть мультиплексора в схеме не используется. Для размещения графического образа на поле чертежа применяется команда *Component/Digital Library/74xx120-/153-/74153*. Установка каких-либо параметров в диалоговом окне свойств мультиплексора не требуется – он уже ассоциирован со своей математической моделью. На графическом образе мультиплексора:

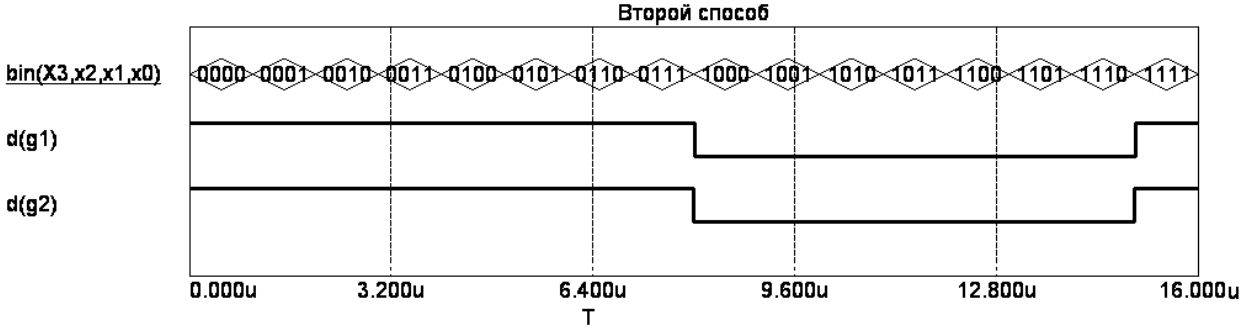
- группы 1C0…1C3 и 2C0…2C3 – информационные входы первой и второй независимых частей мультиплексора;

- A, B – адресные входы, причем A – младший разряд;

- инверсные входы 1GBAR и 2GBAR – входы разрешения работы для первой и второй частей мультиплексора;

- выходы 1Y и 2Y – выходы первой и второй частей мультиплексора.

Временные зависимости выходного сигнала для двух схем представлены на рисунке 3.10. Входной сигнал изображен в виде четырехразрядной шины $x_3x_2x_1x_0$. Сравнительный анализ выходных сигналов в двух схемах показывает их совпадение.



$d(g1)$ – на основе УЛМ; $d(g2)$ – на основе проверочной схемы

Рисунок 3.10 – Временная диаграмма функции $g(x_3, x_2, x_1, x_0)$

III этап. Синтез логического устройства на основе пирамидальной структуры УЛМ.

При выполнении этапа исследования использованы приемы №1 – 7 раздела «Типовые приемы работы в MicroCAP…».

По условию логическая функция пяти аргументов $h(x_4, x_3, x_2, x_1, x_0)$ должна быть аппаратно реализована в виде двухъярусной пирамидальной структуры УЛМ. Для таких структур основным уравнением является:

$$n = k + p,$$

где n – количество аргументов исходной функции; k – число аргументов, подаваемых на мультиплексор второго яруса; p – число аргументов, от которых зависят остаточные функции, воспроизводимые мультиплексорами первого яруса.

Вообще числа k и p могут быть выбраны произвольно, однако напомним, что следует стремиться к *разумным* минимальным значениям k. Зададимся значением $k = 2$, тогда $p = n – k = 3$. Учтем также, что каноническое решение (пирамидальная структура) предполагает подачу на мультиплексоры второго яруса аргументов, играющих роль младших разрядов (см. п. 3.3). Этой информации достаточно, чтобы предложить словесное описание пирамидальной структуры УЛМ для нашего примера.

Во втором ярусе находится мультиплексор типа «4 – 1». Число подаваемых на него аргументов определено нами как $k = 2$; имена аргументов будут x_1x_0; число информационных входов $2^k = 2^2 = 4$.

В первом ярусе находятся четыре мультиплексора типа «8 – 1». Число мультиплексоров первого яруса равно числу информационных входов мультиплексора второго яруса. Число подаваемых аргументов на каждый мультиплексор первого яруса $p = 3$; имена аргументов $x_4x_3x_2$; количество информационных входов каждого мультиплексора $2^p = 2^3 = 8$.

Сигналы настройки для мультиплексоров первого яруса найдем с помощью разложения по Шеннону. Согласно приведенному выше описанию во втором ярусе УЛМ действует два аргумента x_1x_0. Значит, требуется провести разложение по этим двум аргументам, чтобы получить четыре остаточные функции для мультиплексоров первого яруса.

$$h(x_4, x_3, x_2, x_1, x_0) = x_2x_0 + x_4\bar{x}_3x_1 = \bar{x}_1\bar{x}_0 \cdot (x_2 \cdot 0 + x_4\bar{x}_3 \cdot 0) +$$
$$+ \bar{x}_1x_0 \cdot (x_2 \cdot 1 + x_4\bar{x}_3 \cdot 0) + x_1\bar{x}_0 \cdot (x_2 \cdot 0 + x_4\bar{x}_3 \cdot 1) + x_1x_0 \cdot (x_2 \cdot 1 + x_4\bar{x}_3 \cdot 1) =$$
$$= \bar{x}_1\bar{x}_0 \cdot h_0 + \bar{x}_1x_0 \cdot h_1 + x_1\bar{x}_0 \cdot h_2 + x_1x_0 \cdot h_3.$$

Внимание! Порядок следования наборов аргументов, по которым ведется разложение, имеет большое значение. Наборы аргументов должны образовывать возрастающую последовательность двоичных чисел, символическую запись которых отражают наборы. Для нашего примера порядок следования наборов аргументов $\bar{x}_1\bar{x}_0$, $\bar{x}_1x_0$, $x_1\bar{x}_0$, x_1x_0 в разложении соответствует естественному порядку счета $(00)_2$, $(01)_2$, $(10)_2$, $(11)_2$.

Представим в явном и отдельном виде остаточные функции:

$$\begin{cases} h_0(x_4, x_3, x_2) = 0, \\ h_1(x_4, x_3, x_2) = x_2, \\ h_2(x_4, x_3, x_2) = x_4\bar{x}_3, \\ h_3(x_4, x_3, x_2) = x_2 + x_4\bar{x}_3. \end{cases}$$

Настройка мультиплексора, воспроизводящего остаточную функцию h_0, не представляет затруднений. На все информационные входы этого мультиплексора подается логический ноль.

Остаточную функцию $h_1(x_4, x_3, x_2) = x_2$ представим в СДНФ.

$$h_2(x_4, x_3, x_2) = x_2 = x_2(x_3 + \bar{x}_3)(x_4 + \bar{x}_4) = (x_3x_2 + \bar{x}_3x_2)(x_4 + \bar{x}_4) =$$
$$= x_4x_3x_2 + x_4\bar{x}_3x_2 + \bar{x}_4x_3x_2 + \bar{x}_4\bar{x}_3x_2.$$

Наборы аргументов СДНФ отражают следующие числа:

$$x_4x_3x_2 \Rightarrow (111)_2 = (7)_{10};$$
$$x_4\bar{x}_3x_2 \Rightarrow (101)_2 = (5)_{10};$$
$$\bar{x}_4x_3x_2 \Rightarrow (011)_2 = (3)_{10};$$
$$\bar{x}_4\bar{x}_3x_2 \Rightarrow (001)_2 = (1)_{10}.$$

Значит, для мультиплексора, воспроизводящего функцию h_1, на входы D1, D3, D5 и D7 подается логическая единица, на остальные входы – логический ноль.

Аналогично для остаточной функции $h_2(x_4, x_3, x_2) = x_4\bar{x}_3$ имеем:

$$h_2(x_4, x_3, x_2) = x_4\bar{x}_3 = x_4\bar{x}_3(x_2 + \bar{x}_2) = x_4\bar{x}_3x_2 + x_4\bar{x}_3\bar{x}_2$$

Наборы аргументов СДНФ:

$$x_4\bar{x}_3x_2 \Rightarrow (101)_2 = (5)_{10};$$
$$x_4\bar{x}_3\bar{x}_2 \Rightarrow (100)_2 = (4)_{10}.$$

Для мультиплексора, воспроизводящего функцию h_2, на входы D4 и D5 подается логическая единица, на остальные входы – логический ноль.

В случае остаточной функции $h_3(x_4, x_3, x_2) = x_2 + x_4\bar{x}_3$ имеем:

$$h_3(x_4, x_3, x_2) = x_2 + x_4\bar{x}_3 = x_2(x_4 + \bar{x}_4)(x_3 + \bar{x}_3) + x_4\bar{x}_3(x_2 + \bar{x}_2) =$$
$$= (x_4x_2 + \bar{x}_4x_2)(x_3 + \bar{x}_3) + x_4\bar{x}_3x_2 + x_4\bar{x}_3\bar{x}_2 =$$
$$= x_4x_3x_2 + \bar{x}_4x_3x_2 + x_4\bar{x}_3x_2 + \bar{x}_4\bar{x}_3x_2 + x_4\bar{x}_3x_2 + x_4\bar{x}_3\bar{x}_2 =$$
$$= x_4x_3x_2 + \bar{x}_4x_3x_2 + x_4\bar{x}_3x_2 + \bar{x}_4\bar{x}_3x_2 + x_4\bar{x}_3\bar{x}_2.$$

Наборы аргументов СДНФ:

$$x_4x_3x_2 \Rightarrow (111)_2 = (7)_{10};$$

$$\bar{x}_4 x_3 x_2 \Rightarrow (011)_2 = (3)_{10};$$
$$x_4 \bar{x}_3 x_2 \Rightarrow (101)_2 = (5)_{10};$$
$$\bar{x}_4 \bar{x}_3 x_2 \Rightarrow (001)_2 = (1)_{10};$$
$$x_4 \bar{x}_3 \bar{x}_2 \Rightarrow (100)_2 = (4)_{10}.$$

Для мультиплексора, воспроизводящего функцию h_3, на входы D1, D3, D4, D5, D7 подается логическая единица, на остальные входы – логический ноль.

На основе проведенного анализа пирамидальной двухъярусной структуры УЛМ в программе MicroCAP составлена структурная схема логического устройства (рисунок 3.11), реализующего функцию $h(x_4, x_3, x_2, x_1, x_0)$. Проверочная схема устройства, составленная непосредственно по выражению (3.3), приведена отдельно на рисунке 3.12.

В первом ярусе пирамидального УЛМ использованы мультиплексоры 74151A; во втором ярусе задействована верхняя часть сдвоенного мультиплексора 74153. Наличие в схеме пяти аргументов потребовало применения восьмиразрядного цифрового источника сигнала Stim8. Пятиразрядного источника в программе MicroCAP не предусмотрено.

Установка параметров для источника Stim8 несколько отличается от аналогичных действий для Stim4. В диалоговом окне свойств Stim8 в строке FORMAT следует указать значение 44. Две четверки трактуются как две тетрады с шестнадцатеричной ($2^4 = 16$) системой представления. Очевидно, что две тетрады образуют байт с диапазоном значений $[0, FF]_{16}$. Для нашего примера используется только часть диапазона $[0, 1F]_{16}$. При выборе строки COMMAND следует записать короткий программный листинг в нижней части диалогового окна:

```
.DEFINE INPUT
+ 0us 00
+ LABEL begin
+ +1us INCR BY 01
+ +1us GOTO begin -1 TIMES
```

Особое внимание при разработке схемы следует обратить на строгое соответствие старшинства разрядов аргументов и старшинства адресных входов мультиплексора. В пирамидальной структуре аргументы располагаются в порядке убывания старшинства как x_4, x_3, x_2, x_1, x_0. Старшинство адресных входов мультиплексора убывает как C, B, A для первого яруса и B, A – для второго.

Логическая функция пяти аргументов $h(x_4, x_3, x_2, x_1, x_0)$ для отображения всех своих значений требует $2^5 = 32$ временных интервала (цикла). Общая длительность временного анализа схемы составляет $32 \cdot T$; где T – длительность одного цикла. Если в диалоговом окне свойств источника Stim8 было определено, что $T = 1$ мкс, то в строке ввода Time Range диалогового окна Transient Analysis Limits следует указать 32u.

Состояния пятиразрядной шины аргументов $x_4x_3x_2x_1x_0$ представлены на графике в виде десятичных чисел (рисунок 3.13). Из рисунка видно, что временные зависимости выходных сигналов для двух схем взаимно совпадают, значит III этап исследования проведен адекватно.

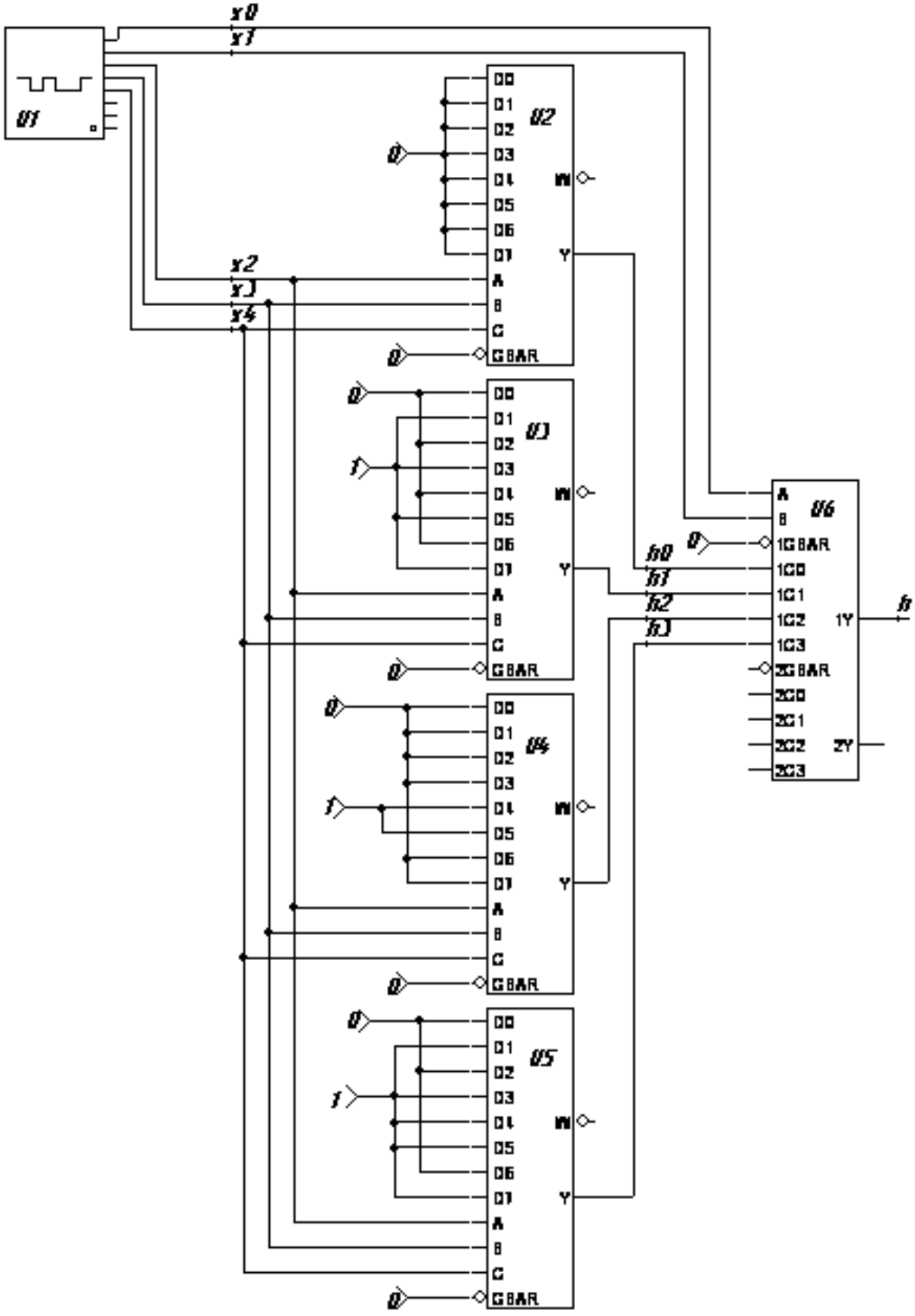


Рисунок 3.11 – Структурная схема логического устройства
по третьему способу настройки

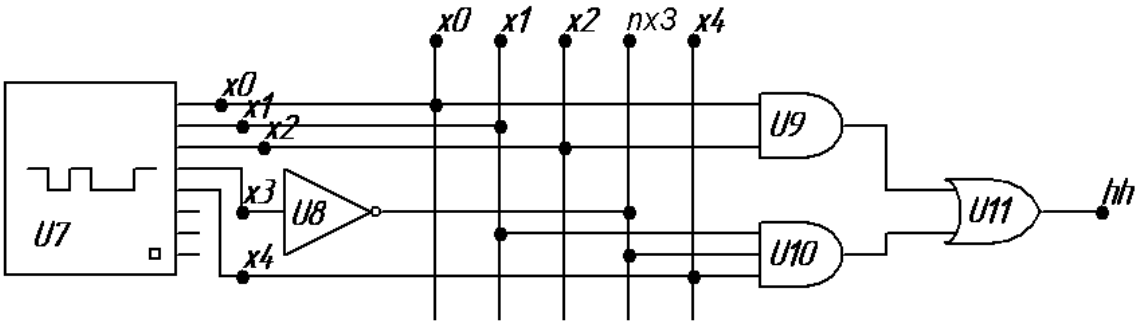


Рисунок 3.12 – Проверочная схема логического устройства
по третьему способу настройки

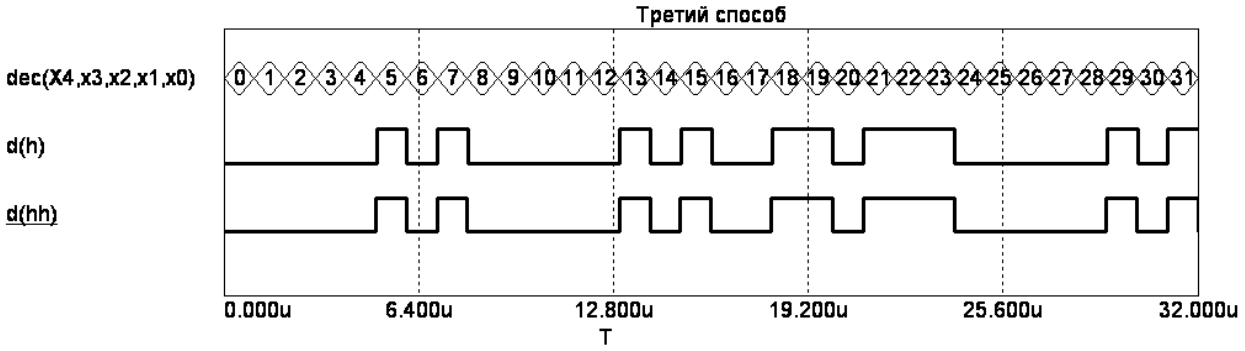


Рисунок 3.13 – Временная диаграмма функции $h(x_4, x_3, x_2, x_1, x_0)$:
$d(h)$ – на основе УЛМ; $d(hh)$ – на основе проверочной схемы

3.5 Лабораторное задание

Повторить методический пример, приведенный выше, по исходным данным Вашего варианта. *Критерием правильности* проведенного исследования является совпадение временных зависимостей выходного сигнала для структурной и проверочной схемы на I, II и III этапах.

3.6 Контрольные вопросы

1. Сравнение структурных (на основе УЛМ) и проверочных схем показывает, что проверочные схемы гораздо проще. В чем тогда заключается преимущество аппаратной реализации логической функции с помощью УЛМ?
2. Что подается на входы мультиплексора при использовании его в качестве УЛМ?
3. Что такое алфавит настройки?
4. Что такое литерал?
5. Какой именно аргумент целесообразно переносить в сигналы настройки УЛМ?
6. Каково правило уменьшения аппаратных затрат для случая пирамидальной двухъярусной структуры УЛМ?
7. Что такое разложение по Шеннону?
8. Что такое остаточная функция?

3.7 Варианты заданий

Таблица 3.8 – Варианты задания для первого способа настройки УЛМ

Номер варианта	Логическая функция $f(x_2, x_1, x_0)$
1	$\overline{x}_0 + x_2\overline{x}_1$
2	$x_1x_0 + x_2x_1 + x_2x_0$
3	$x_1x_0 + \overline{x}_2\overline{x}_1 + x_2x_0$
4	$x_2x_1x_0 + \overline{x}_2\overline{x}_1\overline{x}_0$
5	$\overline{x}_1x_0 + x_2x_0$
6	$x_2 + x_1 + x_0$
7	$x_2 + \overline{x}_1 + x_0$
8	$\overline{x}_1\overline{x}_0 + x_2x_1$
9	$x_2\overline{x}_0 + \overline{x}_2x_1$
10	$\overline{x}_0 + \overline{x}_2x_1$
11	$x_2 + x_1 + \overline{x}_0$
12	$x_1x_0 + \overline{x}_2\overline{x}_1 + \overline{x}_2x_0$
13	$x_2x_1 + \overline{x}_1\overline{x}_0 + \overline{x}_2\overline{x}_1$
14	$x_2\overline{x}_1x_0 + \overline{x}_2x_1\overline{x}_0$
15	$\overline{x}_2 + x_1 + x_0$
16	$x_2x_1 + x_2\overline{x}_0$
17	$\overline{x}_2x_1x_0 + x_2\overline{x}_1\overline{x}_0$
18	$\overline{x}_2 + \overline{x}_1 + \overline{x}_0$
19	$x_1\overline{x}_0 + \overline{x}_2x_1 + x_2x_0$
20	$\overline{x}_2 + \overline{x}_1\overline{x}_0$
21	$\overline{x}_2x_1 + x_1\overline{x}_0 + \overline{x}_2x_0$
22	$\overline{x}_2x_1\overline{x}_0 + x_2\overline{x}_1\overline{x}_0$
23	$\overline{x}_2\overline{x}_1 + \overline{x}_1\overline{x}_0$
24	$x_2 + \overline{x}_1 + \overline{x}_0$
25	$x_2 + \overline{x}_1x_0$

Таблица 3.9 – Варианты задания для второго способа настройки УЛМ

Номер варианта	Логическая функция $g(x_3, x_2, x_1, x_0)$
1	$x_1x_0 + x_2x_0 + x_3$
2	$x_0 + x_1 + \overline{x}_2 + x_3$
3	$x_2\overline{x}_1x_0 + x_3$
4	$x_3\overline{x}_2\overline{x}_1x_0$
5	$x_0 + \overline{x}_1 + x_2 + \overline{x}_3$
6	$x_1x_0 + x_3x_2$
7	$\overline{x}_1x_0 + x_3\overline{x}_2$
8	$x_0 + x_2x_1 + x_3x_2$
9	$x_0 + \overline{x}_2\overline{x}_1 + x_3x_2$
10	$x_2x_1x_0 + x_3x_2x_1$
11	$x_2x_1x_0 + \overline{x}_3\overline{x}_2\overline{x}_1$
12	$x_2\overline{x}_1x_0 + \overline{x}_3x_2\overline{x}_1$
13	$x_3x_2x_1x_0 + x_3\overline{x}_2\overline{x}_1x_0$
14	$x_1\overline{x}_0 + x_3x_2$
15	$x_0 + \overline{x}_2x_1 + \overline{x}_3$
16	$x_1x_0 + x_2x_0 + x_3x_0$
17	$x_1\overline{x}_0 + \overline{x}_2x_0 + x_3\overline{x}_0$
18	$x_2x_1x_0 + x_3x_1x_0$
19	$\overline{x}_0 + \overline{x}_1 + \overline{x}_2 + \overline{x}_3$
20	$x_1x_0 + \overline{x}_2\overline{x}_0 + x_3$
21	$\overline{x}_3 + x_2 + x_1 + \overline{x}_0$
22	$x_3\overline{x}_2x_0 + \overline{x}_1$
23	$\overline{x}_3x_2x_1\overline{x}_0$
24	$x_3\overline{x}_0 + x_2\overline{x}_1$
25	$x_3\overline{x}_1x_0 + x_2x_1\overline{x}_0$

Таблица 3.10 – Варианты задания для третьего способа настройки УЛМ

Номер варианта	Логическая функция $h(x_4, x_3, x_2, x_1, x_0)$
1	$x_1x_0 + x_4x_2x_0 + x_3$
2	$x_0 + x_1 + \overline{x}_2 + x_3 + x_4$
3	$x_2\overline{x}_1x_0 + x_4x_3$
4	$x_3x_2x_1x_0 + \overline{x}_4$
5	$x_0 + \overline{x}_1 + x_2 + \overline{x}_3 + x_4$
6	$x_1x_0 + x_3x_2 + x_4$
7	$\overline{x}_1x_0 + x_3\overline{x}_2 + x_4$
8	$x_0 + x_2x_1 + x_4x_3x_2$
9	$x_0 + \overline{x}_2\overline{x}_1 + x_4x_3x_2$
10	$x_2x_1x_0 + x_3x_2x_1 + x_4x_3x_2$
11	$x_2x_1x_0 + \overline{x}_3\overline{x}_2\overline{x}_1 + x_4x_3x_2$
12	$x_2\overline{x}_1x_0 + \overline{x}_3x_2\overline{x}_1 + x_4\overline{x}_3x_2$
13	$x_3x_2x_1x_0 + \overline{x}_4x_2\overline{x}_1x_0$
14	$x_1\overline{x}_0 + x_3x_2 + \overline{x}_4$
15	$x_0 + \overline{x}_2x_1 + x_4\overline{x}_3$
16	$x_4x_0 + x_1x_0 + x_2x_0 + x_3x_0$
17	$\overline{x}_4x_0 + x_1\overline{x}_0 + \overline{x}_2x_0 + x_3\overline{x}_0$
18	$x_2x_1x_0 + x_3x_1x_0 + x_4x_1x_0$
19	$\overline{x}_0 + \overline{x}_1 + \overline{x}_2 + \overline{x}_3 + \overline{x}_4$
20	$\overline{x}_4x_1 + x_3\overline{x}_2x_1 + x_0$
21	$\overline{x}_4 + x_3 + \overline{x}_2 + x_1 + \overline{x}_0$
22	$x_4\overline{x}_3\overline{x}_2 + \overline{x}_1x_0$
23	$x_4\overline{x}_3\overline{x}_2x_1 + x_0$
24	$\overline{x}_4\overline{x}_3 + \overline{x}_2\overline{x}_1 + \overline{x}_0$
25	$x_3\overline{x}_2\overline{x}_1x_0 + \overline{x}_4x_2x_1\overline{x}_0$

4 ЛАБОРАТОРНАЯ РАБОТА №4 – ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ АВТОМАТОВ НА JK-ТРИГГЕРАХ

4.1 Цель работы

В ходе выполнения настоящей работы предусматривается:

1) знакомство с особенностями проектирования синхронных автоматов с памятью;

2) исследование структуры автоматов Мура;

3) приобретение навыков двоичного кодирования состояний для синхронного автомата;

4) приобретение навыков по нахождению функций возбуждения для *JK*-триггеров.

4.2 Порядок выполнения работы

1. Изучить методические указания к лабораторной работе.
2. Письменно, в отчете по лабораторной работе ответить на контрольные вопросы.
3. Внимательно ознакомиться с примером, приведенным в пункте 4.4.
4. Выполнить лабораторное задание согласно варианту задания.
5. Сделать выводы по работе.

Внимание! Отчет по лабораторной работе в обязательном порядке должен содержать: схемы включения, графики зависимостей, все необходимые расчеты и их результаты, текстовые пояснения. На графиках в отчете должны присутствовать единицы измерения, масштаб, цена деления.

4.3 Методика проектирования автоматов с памятью

Узлы и устройства, которые содержат элементы памяти, относятся к классу автоматов с памятью (АП). Наличие элементов памяти (ЭП) придает АП свойство иметь некоторое внутреннее состояние Q, определяемое совокупностью состояний всех элементов памяти. В зависимости от внутреннего состояния (далее называемого просто состоянием), АП различно реагирует на один и тот же вектор входных сигналов X. Воспринимая входные сигналы при определенном состоянии, АП переходит в новое состояние и вырабатывает вектор выходных переменных Y. Таким образом, для АП:

$$Q_{\text{Н}} = f(Q, \text{X}),$$
$$\text{Y} = \varphi(Q, \text{X}),$$

где $Q_{\text{Н}}$ и Q – состояния АП после и до подачи входных сигналов (индекс «н» от слова «новое»).

Переходы АП из одного состояния в другое начинаются с некоторого исходного состояния Q_0, задание которого также является частью задания автомата. Следующее состояние зависит от Q_0 и поступивших входных сигналов X. В конечном счете, текущее состояние и выходы автомата зависят от начального состояния и всех векторов X, поступавших на автомат в предшествующих сменах входных сигналов. Таким образом, вся последовательность входных сигналов определяет последовательность состояний и выходных сигналов. Это объясняет название «*последовательностные схемы*», также применяемое для обозначения АП.

Структурно АП отличаются от КЦ наличием в их схемах обратных связей, вследствие чего в них проявляются свойства запоминания состояний (достаточно вспомнить схемы триггерных элементов, где указанная особенность проявляется очень наглядно).

Автоматы с памятью в каноническом представлении разделяют на две части: *память* и *комбинационную цепь*. На входы КЦ подаются входные сигналы и сигналы состояния АП. На ее выходе вырабатываются выходные сигналы и сигналы перевода АП в новое состояние.

Принципиальным является деление АП на *асинхронные* и *синхронные*. В асинхронных (рисунок 4.1) роль элементов памяти играют элементы задержки, через которые сигналы состояния передаются на входы КЦ, чтобы совместно с новым набором входных переменных определить следующую пару значений Y и *Q* на выходе. Элементы АП переключаются здесь под непосредственным воздействием изменений информационных сигналов. Скорость распространения процесса переключений в цепях асинхронного автомата определяется собственными задержками элементов.

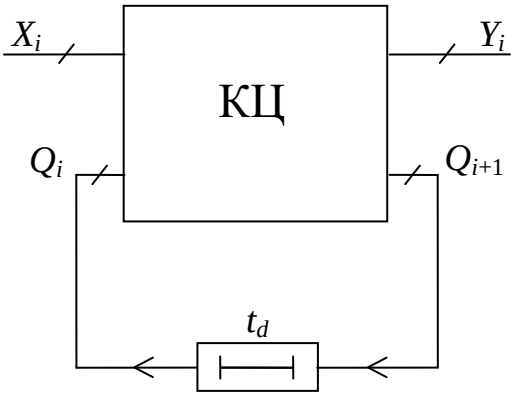


Рисунок 4.1 – Асинхронный автомат с памятью

В синхронном АП (рисунок 4.2) имеются специальные синхросигналы (тактирующие импульсы) *C*, которые разрешают элементам памяти прием данных только в определенные моменты времени. Элементами памяти служат синхронные триггеры. Процесс обработки информации упорядочивается во времени, и в течение одного такта возможно распространение процесса переключения только в строго определенных пределах тракта обработки информации.

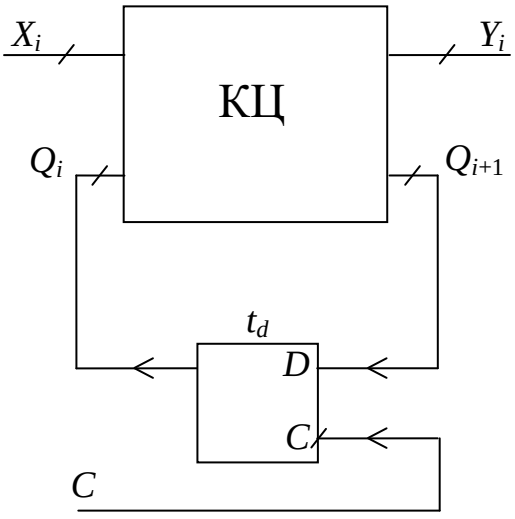


Рисунок 4.2 – Синхронный автомат с памятью

Практическое применение асинхронных автоматов существенно затруднено сильным влиянием на их работу задержек сигналов в цепях АП, создающих статические и динамические риски, гонки элементов памяти (неодновременность срабатывания ЭП даже при одновременной подаче на них входных сигналов) и др. В итоге характерным свойством асин-

хронного автомата является то, что при переходе из одного устойчивого состояния в другое он обычно проходит через промежуточные нестабильные состояния. Нельзя сказать, что методы борьбы с нежелательными последствиями рисков и гонок в асинхронных АП отсутствуют, но все же обеспечение предсказуемого поведения АП – сложная проблема. В сложных АП асинхронные схемы встречаются редко, а в простейших схемах применяются. Примером могут служить асинхронные *RS*-триггеры.

В синхронных автоматах каждое новое состояние устойчиво и переходные временные состояния не возникают. Концепция борьбы с последствиями рисков и гонок в синхронных автоматах проста – прием информации в элементы памяти разрешается только после завершения в схеме переходных процессов. Это обеспечивается параметрами синхроимпульсов, задающих интервалы времени для завершения тех или иных процессов. В сравнении с асинхронными, синхронные АП значительно проще в проектировании.

На сегодняшний день и достаточно длительную перспективу основным путем построения АП следует считать применение тактирования, т.е. синхронных автоматов.

В работах отечественных и зарубежных ученых разрабатывается направление, называемое проектирование самосинхронизирующихся устройств, в которых тактовые импульсы следуют с переменной частотой, зависящей от длительности реального переходного процесса в схеме. Однако перспективность этого направления еще не вполне ясна.

В теории автоматов проводится их классификация по ряду признаков. В схемотехнике преобладают автоматы Мура, выходы которых являются функциями только состояния автомата. Для этого автомата:

$$Q_{\text{H}} = f(Q, \text{X}),$$
$$\text{Y} = \varphi(Q).$$

Зависимость выходов и от состояния автомата и от вектора входных переменных свойственна автоматам Мили.

Некоторые функциональные узлы принадлежат к числу автономных автоматов, которые не имеют информационных входов, и под действием тактовых сигналов переходят из состояния в состояние по алгоритму, определяемому структурой автомата.

Проектирование автоматов. Проектирование АП содержит следующие этапы:

- исходное задание функционирования;
- формализованное задание функционирования;
- минимизация состояний;
- кодирование состояний;
- составление таблицы переходов;
- определение функций возбуждения элементов памяти (триггеров);
- минимизация функций возбуждения триггеров;
- переход к базису выбранной для реализации схемотехнологии;
- составление логической схемы;
- сборка и проверка автомата.

Исходное задание функционирования может иметь различную форму, в том числе и словесную. От нее переходят к формализованному заданию – таблицам, формулам, диаграммам состояния и т.п. Далее выполняются минимизация и кодирование состояний автомата, в результате чего получается таблица переходов, на основании которой можно найти функции возбуждения триггеров.

Минимизация и кодирование состояний в общем случае задача, решение которой может потребовать значительных усилий, однако при проектировании узлов ЭВМ и цифровой автоматики она чаще всего проста, и ее решение подсказывается самой формулировкой задания на проектирование. Традиционно *широко применяется кодирование состояний автомата двоичными кодами*, при котором триггеры используются в схеме экономно. Для некоторых новых СБИС программируемой логики, снабженных большим числом триггеров, экономия их числа при построении автомата несущественна. Для таких случаев применение ко-

дирования кодами «1 из N» может быть предпочтительным, т.к. приводит к более быстродействующим схемам, хотя и требующим значительного числа триггеров.

Функции возбуждения триггеров, обеспечивающие переходы АП из одного состояния в другое, реализуются его комбинационной частью. Они, как сказано в перечислении этапов проектирования, минимизируются и переводятся в базис выбранных средств реализации автомата. Это положение следует понимать в широком смысле, поскольку в зависимости от средств реализации КЦ требования к формам представления функций возбуждения могут существенно различаться. Точнее можно говорить о приведении функций к виду, удобному для воспроизведения данными средствами.

После выполнения указанных действий можно получить логическую схему АП. Заканчивается процесс проверкой работы узла с помощью моделирования или макетирования.

Рассмотрим более подробно методику проектирования автоматов, содержащих триггеры (рисунок 4.3).

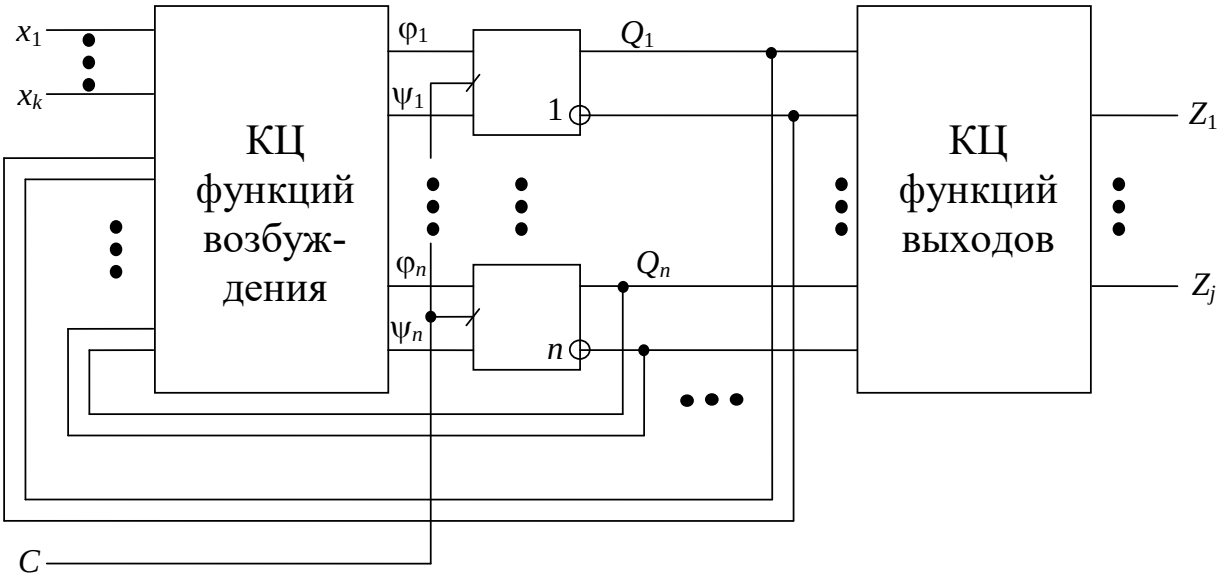


Рисунок 4.3 – Структурная схема автомата Мура

В тактируемых автоматах элементами памяти служат синхронные триггеры, причем любой автомат можно построить на любом типе триггера (*D, JK, RS, T* и др.).

При двоичном кодировании состояний автомата число триггеров в его схеме равно:

$$n = \lceil \log_2 N \rceil,$$

где N – число состояний автомата; $\lceil\ \rceil$ – знак округления до ближайшего справа целого числа.

При кодировании кодом «1 из N» число триггеров равно числу состояний автомата $n = N$, т.к. каждому состоянию соответствует один триггер в единичном состоянии при нулевом состоянии остальных.

Будем считать, что закон функционирования автомата определен, и кодирование его состояний произведено. Значит, известна последовательность состояний триггеров, принимаемых ими в каждом такте под управлением входных сигналов $x_1, x_2, \ldots, x_k$ и текущего состояния $Q_1, Q_2, \ldots, Q_n$. Предмет синтеза – получение функций возбуждения φ_i и ψ_i для каждого входа всех триггеров, обеспечивающих необходимые переходы автоматов.

Функции выхода для автоматов Мура зависят только от состояния автомата, поэтому нахождение выходов $Z_1, Z_2, \ldots, Z_j$ осуществляется комбинационной схемой, на которую подаются только выходы триггеров $Q_1, Q_2, \ldots, Q_n$. Все триггеры тактируются общим синхросигналом C. После завершения выработки функций возбуждения комбинационной схемой поступает очередной тактовый сигнал C, переводящий триггеры в новое состояние.

Вид функций возбуждения зависит от логического типа триггеров. Поэтому одним из средств синтеза служат «словари» для триггеров.

При поиске функций возбуждения триггеров вначале составляется таблица (таблица 4.1), содержащая приведенные ниже данные.

Таблица 4.1 – Функции возбуждения триггеров

Входы в момент времени t				Состояния триггеров								Необходимые сигналы на всех входах каждого триггера				
				Q (старое)				$Q_н$ (новое)								
X_1	X_2	…	X_k	Q_1	Q_2	…	Q_n	Q_1	Q_2	…	Q_n	φ_1	ψ_1	…	φ_n	ψ_n

Столбцы φ_1, ψ_1, …, φ_n, ψ_n определяют функции возбуждения триггеров.

Многовариантность реализаций автомата связана с выбором типа триггеров и комбинационной части.

Относительно наиболее распространенных типов триггеров *JK* и *D* можно отметить следующее. Триггер типа *JK* обладает более развитыми логическими функциями, поэтому для него функции возбуждения в среднем более просты, но число их вдвое больше, чем для триггера *D*. Что же даст более простое решение, заранее неизвестно.

Комбинационная часть автомата может быть построена на логических элементах, мультиплексорах, ИС программируемой памяти, программируемых логических матрицах и т.д.

Состояния автомата можно кодировать двоичными кодами, кодами «1 из *N*» и др.

Автомат можно построить, приспособив к необходимому функционированию типовую ИС среднего уровня интеграции (счетчик, сдвигающий регистр), добавив к ним специально спроектированную логическую часть.

4.4 Пример проектирования автомата на основе *JK*-триггеров

Требуется спроектировать трехразрядный автомат на основе *JK*-триггеров с двумя режимами работы, управляемый входным сигналом *M*. При $M = 0$ автомат должен работать как двоичный счетчик с модулем счета 8, а при $M = 1$ – как счетчик в коде Грея. Комбинационные цепи автомата реализовать в базисе И-НЕ. Частота тактовых импульсов синхронизации 100 кГц. Правильность предложенного схемотехнического решения подтвердить результатами моделирования в программе MicroCAP.

Код Грея используется в системах контроля цифровых устройств, преобразователях механических перемещений в цифровой код и т.д. При переходе от предыдущей кодовой комбинации к следующей в коде Грея изменяется только один разряд. На рисунке 4.4 представлена диаграмма состояний трехразрядного кода Грея.

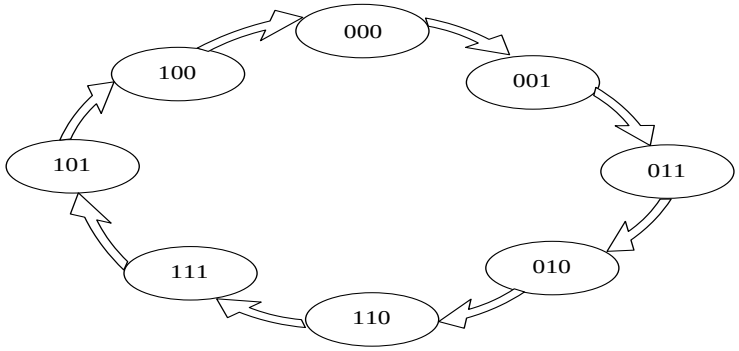


Рисунок 4.4 – Диаграмма состояний трехразрядного кода Грея

I этап. Описание работы цифрового автомата.

На первом этапе проектирования цифрового автомата целесообразно представить его функционирование с помощью диаграммы состояний. Преимущества такого способа представления – наглядность и простота первоначального восприятия. Условимся, что переходы автомата на диаграмме состояний будут обозначены следующим образом:

- для режима двоичного счетчика как $\overline{M}$, что соответствует $M = 0$;
- для режима счетчика в коде Грея как M, что соответствует $M = 1$.

Восемь возможных состояний ($2^3 = 8$) цифрового автомата следует расположить на диаграмме в естественном порядке их наступления, характерном для какого-либо режима. Пусть базовая последовательность состояний на диаграмме соответствует режиму двоичного счетчика. Тогда дополнив рисунок переходами, отражающими последовательность кода Грея, получим полную диаграмму состояний двухрежимного цифрового автомата (рисунок 4.5).

Используя понятия теории графов, можно сказать, что представленная диаграмма состояний – это направленный граф, т.к. направления переходов заданы. Каждый переход на диаграмме или ребро графа соединяет старое (предыдущее) состояние автомата и новое (последующее) состояние; причем новое состояние в такой паре всегда указано стрелкой. Перебор всех ребер графа и анализ состояний позволяет составить таблицу истинности цифрового автомата для двух режимов функционирования (таблица 4.2).

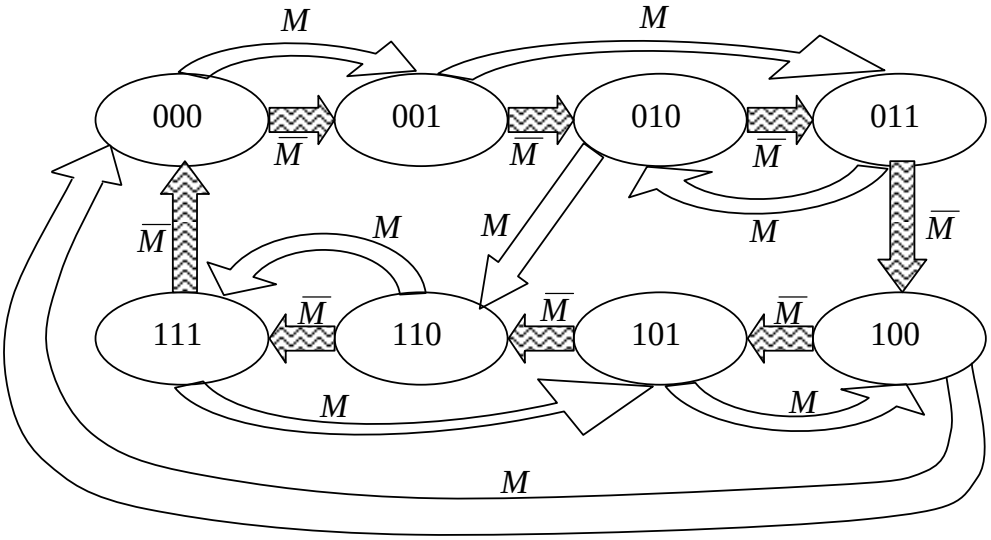


Рисунок 4.5 – Диаграмма состояний двухрежимного трехразрядного автомата

Правило заполнения таблицы истинности. Для каждого режима заполнение таблицы истинности следует начинать с блока Q_2, Q_1, Q_0 перечислением кодовых комбинаций аргументов в порядке естественного счета двоичной системы счисления: 000, 001, …, 111. Затем, учитывая, что кодовые комбинации в блоке Q_2, Q_1, Q_0 представляют собой старые значения состояний автомата, необходимо найти для каждого такого состояния по диаграмме соответствующее новое значение и записать правее в ячейки Q_{2H}, Q_{1H}, Q_{0H}. Бессистемное (произвольное) расположение пар состояний в таблице истинности оказывается критичным для некоторых способов синтеза цифровых автоматов, например, для синтеза автоматов с мультиплексным управлением.

Таблица 4.2 – Таблица истинности двухрежимного цифрового автомата

Входной управляющий сигнал	Исходное состояние			Новое состояние		
M	Q_2	Q_1	Q_0	$Q_{2н}$	$Q_{1н}$	$Q_{0н}$
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	1	1	0
0	1	1	0	1	1	1
0	1	1	1	0	0	0
1	0	0	0	0	0	1
1	0	0	1	0	1	1
1	0	1	0	1	1	0
1	0	1	1	0	1	0
1	1	0	0	0	0	0
1	1	0	1	1	0	0
1	1	1	0	1	1	1
1	1	1	1	1	0	1

II этап. Составление функций переходов JK-триггеров.

При выполнении этапа исследования использован прием №8 раздела «Типовые приемы работы в MicroCAP…».

В таблице истинности (таблица 4.2) новые состояния автомата $Q_{2Н}$, $Q_{1Н}$, $Q_{0Н}$ являются значениями трех логических функций, а входной управляющий сигнал M и старые состояния Q_2, Q_1, Q_0 можно рассматривать как аргументы логических функций:

$$\begin{aligned} Q_{2Н} &= f(M, Q_2, Q_1, Q_0); \\ Q_{1Н} &= g(M, Q_2, Q_1, Q_0); \\ Q_{0Н} &= h(M, Q_2, Q_1, Q_0). \end{aligned} \tag{4.1}$$

Пользуясь таблицей истинности, каждую из функций (4.1) можно записать сначала в СДНФ, а затем упростить до МДНФ. Автоматизируем этот процесс с помощью логического преобразователя Electronics Workbench.

$$\begin{aligned} Q_{2H} &= \bar{M}\cdot\bar{Q}_2Q_1Q_0 + \bar{M}\cdot Q_2\bar{Q}_1\bar{Q}_0 + \bar{M}\cdot Q_2\bar{Q}_1Q_0 + \bar{M}\cdot Q_2Q_1\bar{Q}_0 + \\ &+M\cdot\bar{Q}_2Q_1\bar{Q}_0 + M\cdot Q_2\bar{Q}_1Q_0 + M\cdot Q_2Q_1\bar{Q}_0 + M\cdot Q_2Q_1Q_0 = \\ &= \bar{M}\cdot\bar{Q}_2Q_1Q_0 + \bar{M}\cdot Q_2\bar{Q}_1 + \bar{M}\cdot Q_2\bar{Q}_0 + M\cdot Q_1\bar{Q}_0 + M\cdot Q_2Q_0. \end{aligned} \tag{4.2}$$

$$\begin{aligned} Q_{1H} &= \bar{M}\cdot\bar{Q}_2\bar{Q}_1Q_0 + \bar{M}\cdot\bar{Q}_2Q_1\bar{Q}_0 + \bar{M}\cdot Q_2\bar{Q}_1Q_0 + \bar{M}\cdot Q_2Q_1\bar{Q}_0 + \\ &+M\cdot\bar{Q}_2\bar{Q}_1Q_0 + M\cdot\bar{Q}_2Q_1\bar{Q}_0 + M\cdot\bar{Q}_2Q_1Q_0 + M\cdot Q_2Q_1\bar{Q}_0 = \\ &= \bar{M}\cdot\bar{Q}_1Q_0 + M\cdot\bar{Q}_2Q_0 + Q_1\bar{Q}_0. \end{aligned} \tag{4.3}$$

$$\begin{aligned} Q_{0H} &= \bar{M}\cdot\bar{Q}_2\bar{Q}_1\bar{Q}_0 + \bar{M}\cdot\bar{Q}_2Q_1\bar{Q}_0 + \bar{M}\cdot Q_2\bar{Q}_1\bar{Q}_0 + \bar{M}\cdot Q_2Q_1\bar{Q}_0 + \\ &+M\cdot\bar{Q}_2\bar{Q}_1\bar{Q}_0 + M\cdot\bar{Q}_2\bar{Q}_1Q_0 + M\cdot Q_2Q_1\bar{Q}_0 + M\cdot Q_2Q_1Q_0 = \\ &= \bar{M}\cdot\bar{Q}_0 + M\cdot\bar{Q}_2\bar{Q}_1 + M\cdot Q_2Q_1. \end{aligned} \tag{4.4}$$

III этап. Нахождение функций возбуждения для входов JK-триггеров.

При выполнении этапа исследования использован прием №8 раздела «Типовые приемы работы в MicroCAP…».

Функции переходов триггеров $Q_{i\text{Н}}$ (4.2 – 4.4) выражены, помимо прочих аргументов, также через свое старое состояние $\widetilde{Q}_i$. Это обстоятельство позволяет провести непосредственный сравнительный анализ функций переходов и характеристического уравнения *JK*-триггера, чтобы получить функции возбуждения для входов *J* и *K* каждого триггера. Известно [3], что характеристическое уравнение *JK*-триггера в общем виде записывается как:

$$Q_{iH} = J_i\overline{Q}_i + \overline{K}_i Q_i. \tag{4.5}$$

Значит, для сравнительного анализа необходимо, чтобы функции переходов (4.2 – 4.4) были представлены в виде:

$$Q_{iH} = \alpha_i\overline{Q}_i + \beta_i Q_i, \tag{4.6}$$

где сомножители α_i и β_i уже не содержат переменных $\overline{Q}_i$ и Q_i.

Если воспользоваться разложением Шеннона (см. лабораторную работу №3) по одной переменной, то можно достаточно просто преобразовать функции переходов (4.2 – 4.4) к форме (4.6). В качестве переменной, по которой ведется разложение, выступает старое состояние Q_i.

$$\begin{aligned}
Q_{2H} &= \overline{M}\cdot\overline{Q}_2Q_1Q_0 + \overline{M}\cdot Q_2\overline{Q}_1 + \overline{M}\cdot Q_2\overline{Q}_0 + M\cdot Q_1\overline{Q}_0 + M\cdot Q_2Q_0 = \\
&= \overline{Q}_2\left(\overline{M}\cdot\overline{0}\cdot Q_1Q_0 + \overline{M}\cdot 0\cdot\overline{Q}_1 + \overline{M}\cdot 0\cdot\overline{Q}_0 + M\cdot Q_1\overline{Q}_0 + M\cdot 0\cdot Q_0\right) + \\
&+ Q_2\left(\overline{M}\cdot\overline{1}\cdot Q_1Q_0 + \overline{M}\cdot 1\cdot\overline{Q}_1 + \overline{M}\cdot 1\cdot\overline{Q}_0 + M\cdot Q_1\overline{Q}_0 + M\cdot 1\cdot Q_0\right) = \\
&= \overline{Q}_2\left(\overline{M}\cdot Q_1Q_0 + M\cdot Q_1\overline{Q}_0\right) + Q_2\left(\overline{M}\cdot\overline{Q}_1 + \overline{M}\cdot\overline{Q}_0 + M\cdot Q_1\overline{Q}_0 + M\cdot Q_0\right) = \\
&= \overline{Q}_2\left(\overline{M}\cdot Q_1Q_0 + M\cdot Q_1\overline{Q}_0\right) + Q_2\left(\overline{M}\cdot\overline{Q}_1 + \overline{M}\cdot\overline{Q}_0 + M\cdot Q_1 + M\cdot Q_0\right).
\end{aligned}$$

$$\begin{aligned}
Q_{1H} &= \overline{M}\cdot\overline{Q}_1Q_0 + M\cdot\overline{Q}_2Q_0 + Q_1\overline{Q}_0 = \\
&= \overline{Q}_1\left(\overline{M}\cdot\overline{0}\cdot Q_0 + M\cdot\overline{Q}_2Q_0 + 0\cdot\overline{Q}_0\right) + Q_1\left(\overline{M}\cdot\overline{1}\cdot Q_0 + M\cdot\overline{Q}_2Q_0 + 1\cdot\overline{Q}_0\right) = \\
&= \overline{Q}_1\left(\overline{M}\cdot Q_0 + M\cdot\overline{Q}_2Q_0\right) + Q_1\left(M\cdot\overline{Q}_2Q_0 + \overline{Q}_0\right) = \\
&= \overline{Q}_1\left(\overline{M}\cdot Q_0 + \overline{Q}_2Q_0\right) + Q_1\left(M\cdot\overline{Q}_2 + \overline{Q}_0\right).
\end{aligned}$$

$$\begin{aligned}
Q_{0H} &= \overline{M}\cdot\overline{Q}_0 + M\cdot\overline{Q}_2\overline{Q}_1 + M\cdot Q_2Q_1 = \\
&= \overline{Q}_0\left(\overline{M}\cdot\overline{0} + M\cdot\overline{Q}_2\overline{Q}_1 + M\cdot Q_2Q_1\right) + Q_0\left(\overline{M}\cdot\overline{1} + M\cdot\overline{Q}_2\overline{Q}_1 + M\cdot Q_2Q_1\right) = \\
&= \overline{Q}_0\left(\overline{M} + M\cdot\overline{Q}_2\overline{Q}_1 + M\cdot Q_2Q_1\right) + Q_0\left(M\cdot\overline{Q}_2\overline{Q}_1 + M\cdot Q_2Q_1\right) = \\
&= \overline{Q}_0\left(\overline{M} + \overline{Q}_2\overline{Q}_1 + Q_2Q_1\right) + Q_0\left(M\cdot\overline{Q}_2\overline{Q}_1 + M\cdot Q_2Q_1\right).
\end{aligned}$$

Сравнивая преобразованные функции переходов $Q_{2\text{Н}}$, $Q_{1\text{Н}}$, $Q_{0\text{Н}}$ с характеристическим уравнением (4.5), находим функции возбуждения для входов J_i и K_i:

$$J_2 = \overline{M}\cdot Q_1Q_0 + M\cdot Q_1\overline{Q}_0 \tag{4.7}$$

$$K_2 = \overline{\overline{M}\cdot\overline{Q}_1 + \overline{M}\cdot\overline{Q}_0 + M\cdot Q_1 + M\cdot Q_0} \tag{4.8}$$

$$J_1 = \overline{M}\cdot Q_0 + \overline{Q}_2Q_0 \tag{4.9}$$

$$K_1 = \overline{M \cdot \bar{Q}_2 + \bar{Q}_0} \tag{4.10}$$

$$J_0 = \bar{M} + \bar{Q}_2\bar{Q}_1 + Q_2Q_1 \tag{4.11}$$

$$K_0 = \overline{M \cdot \bar{Q}_2\bar{Q}_1 + M \cdot Q_2Q_1} \tag{4.12}$$

Для дальнейшего практического использования удобно, чтобы функции возбуждения входов K_i (4.8, 4.10, 4.12) были представлены в дизъюнктивной нормальной форме (ДНФ). Напомним, что ДНФ называется такая форма представления функции, при которой логическое выражение функции строится в виде дизъюнкции ряда членов, каждый из которых является простой конъюнкцией аргументов или их инверсией. Функции возбуждения входов J_i (4.7, 4.9, 4.11) уже представлены в ДНФ, поэтому преобразования не требуют. Для перехода от форм (4.8, 4.10, 4.12) к ДНФ вновь воспользуемся логическим преобразователем Electronics Workbench.

$$K_2 = \overline{\bar{M} \cdot \bar{Q}_1 + \bar{M} \cdot \bar{Q}_0 + M \cdot Q_1 + M \cdot Q_0} = \bar{M} \cdot Q_1Q_0 + M \cdot \bar{Q}_1\bar{Q}_0. \tag{4.13}$$

$$K_1 = \overline{M \cdot \bar{Q}_2 + \bar{Q}_0} = \bar{M} \cdot Q_0 + Q_2Q_0. \tag{4.14}$$

$$K_0 = \overline{M \cdot \bar{Q}_2\bar{Q}_1 + M \cdot Q_2Q_1} = \bar{M} + \bar{Q}_2Q_1 + Q_2\bar{Q}_1. \tag{4.15}$$

По условию комбинационные цепи автомата должны быть реализованы в базисе И-НЕ. Функции возбуждения триггеров J_i и K_i (4.7, 4.9, 4.11, 4.13 – 4.15) преобразуем к заданному базису. Для этого воспользуемся правилом (см. лабораторную работу №1): для перехода к базису И-НЕ нужно дважды инвертировать все выражение, а затем применить теорему де Моргана $\overline{x + y} = \bar{x} \cdot \bar{y}$.

$$J_2 = \bar{M} \cdot Q_1Q_0 + M \cdot Q_1\bar{Q}_0 = \overline{\overline{\bar{M} \cdot Q_1Q_0 + M \cdot Q_1\bar{Q}_0}} = \overline{\overline{\bar{M} \cdot Q_1Q_0} \cdot \overline{M \cdot Q_1\bar{Q}_0}} \tag{4.16}$$

$$K_2 = \bar{M} \cdot Q_1Q_0 + M \cdot \bar{Q}_1\bar{Q}_0 = \overline{\overline{\bar{M} \cdot Q_1Q_0 + M \cdot \bar{Q}_1\bar{Q}_0}} = \overline{\overline{\bar{M} \cdot Q_1Q_0} \cdot \overline{M \cdot \bar{Q}_1\bar{Q}_0}} \tag{4.17}$$

$$J_1 = \bar{M} \cdot Q_0 + \bar{Q}_2Q_0 = \overline{\overline{\bar{M} \cdot Q_0 + \bar{Q}_2Q_0}} = \overline{\overline{\bar{M} \cdot Q_0} \cdot \overline{\bar{Q}_2Q_0}} \tag{4.18}$$

$$K_1 = \bar{M} \cdot Q_0 + Q_2Q_0 = \overline{\overline{\bar{M} \cdot Q_0 + Q_2Q_0}} = \overline{\overline{\bar{M} \cdot Q_0} \cdot \overline{Q_2Q_0}} \tag{4.19}$$

$$J_0 = \bar{M} + \bar{Q}_2\bar{Q}_1 + Q_2Q_1 = \overline{\overline{\bar{M} + \bar{Q}_2\bar{Q}_1 + Q_2Q_1}} = \overline{M \cdot \overline{\bar{Q}_2\bar{Q}} \cdot \overline{Q_2Q_1}} \tag{4.20}$$

$$K_0 = \bar{M} + \bar{Q}_2Q_1 + Q_2\bar{Q}_1 = \overline{\overline{\bar{M} + \bar{Q}_2Q_1 + Q_2\bar{Q}_1}} = \overline{M \cdot \overline{\bar{Q}_2Q_1} \cdot \overline{Q_2\bar{Q}_1}} \tag{4.21}$$

IV этап. Синтез автомата на JK-триггерах и элементах И-НЕ.

При выполнении этапа исследования использованы приемы №1, 2, 3, 4, 5, 6, 9 раздела «Типовые приемы работы в MicroCAP…».

По результатам выполнения предыдущих этапов исследования подготовлены все необходимые сведения для синтеза автомата. Сформулируем словесное описание. Основой цифрового автомата являются три *JK*-триггера – по количеству разрядов цифрового кода. К каждому из входов J_i, K_i присоединяется комбинационная цепь, реализованная в базисе И-НЕ согласно (4.16 – 4.21). Номер индекса *i* однозначно определяет взаимное соответствие функции возбуждения и входа триггера. Схема автомата должна быть дополнена вспомогательными цепями синхронизации и сброса.

По предложенному словесному описанию на рисунке 4.6 представлена структурная схема цифрового автомата, подготовленная в программе MicroCAP. В схеме *JK*-триггер *X*2, предназначенный для старшего разряда, расположен слева, далее триггеры идут в порядке убывания старшинства разрядов. Элемент И-НЕ *U*2 исполняет роль инвертора для выработки сигнала $\overline{M}$. Чтобы избежать чрезмерной сложности в восприятии схемы, применен скрытый

способ прокладки электрических проводников. Отрезки проводников, принадлежащие одной и той же цепи, имеют одинаковые названия.

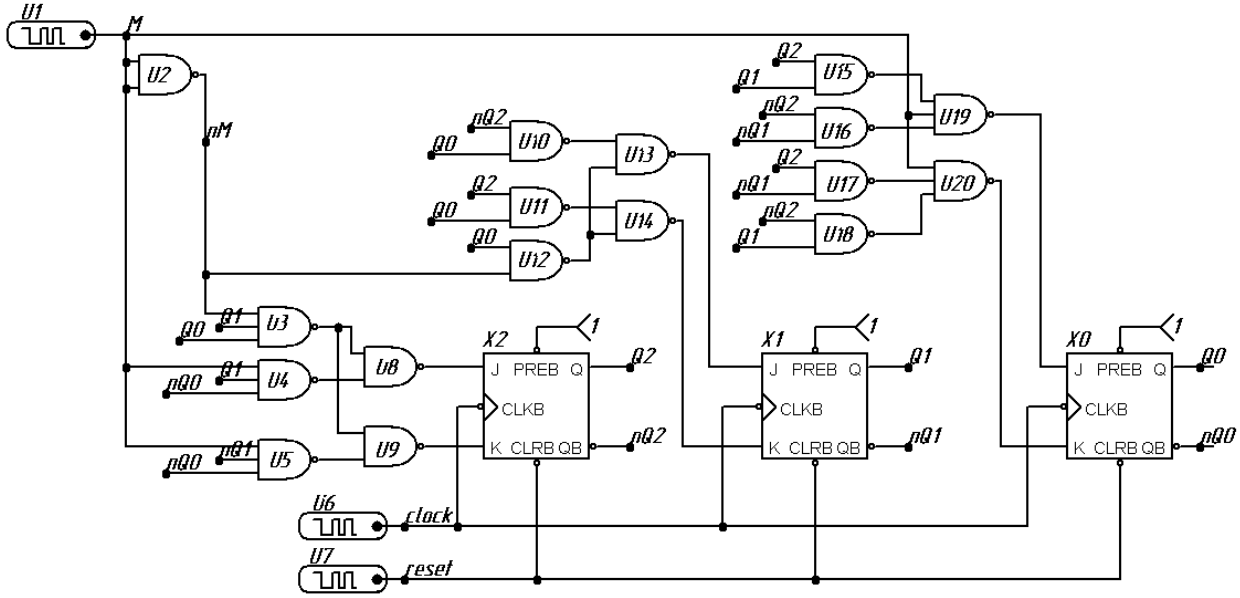


Рисунок 4.6 – Структурная схема цифрового автомата на *JK*-триггерах

Триггеры *X*2, *X*1, *X*0 – синхронные типа JKFF с инверсным динамическим входом, рассмотренные в приеме №9.

Для цифровых источников *U*1, *U*6, *U*7 требуется задать параметры, определяющие поведение сигнала управления, синхросигнала и сигнала сброса, соответственно. Сначала определим длительность временного вида анализа. По условию частота импульсов синхронизации $f = 100$ кГц, значит период следования $T = 10$ мкс, причем в каждом периоде возникает новое состояние автомата. Согласно диаграмме (рисунок 4.5) имеем в общей сложности для двух режимов количество состояний $n = 16$. Общая длительность τ временного анализа:

$$\tau = n \cdot T = 16 \cdot 10 = 160 \text{ мкс}.$$

При исследовании цифрового автомата важно убедиться, что после восьмого (последнего) состояния в каждом режиме он возвращается в исходное положение. Например, по диаграмме (рисунок 4.5) в режиме $M = 0$ за состоянием 111 должно наступать состояние 000; в режиме $M = 1$ за состоянием 100 должно наступать состояние 000. Для контроля этих переходов необходимо добавить еще по одному периоду следования синхроимпульса в каждом режиме. Окончательная длительность $\tau_{ОК}$ временного анализа:

$$\tau_{ОК} = \tau + 2T = 160 + 2 \cdot 10 = 180 \text{ мкс}.$$

Установим параметры синхросигнала *Clock*. Пусть начало периода синхросигнала сопровождается нулевым уровнем, через половину периода $T/2 = 5$ мкс возникает передний фронт импульса и, как следствие, единичный уровень. Приход еще через 5 мкс следующего периода с нулевым уровнем сформирует отрицательный фронт импульса (рисунок 4.7).

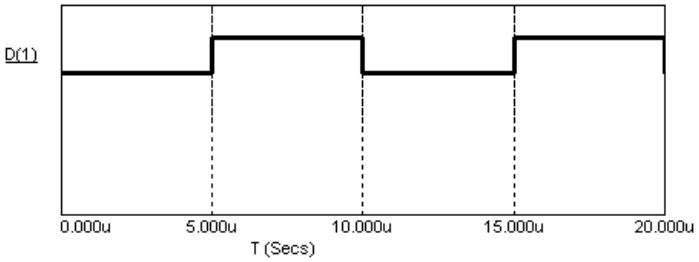


Рисунок 4.7 – Временная диаграмма синхросигнала

В диалоговом окне свойств источника *U*6 в строке FORMAT указывают значение 1. При выборе строки COMMAND задают форму синхросигнала в нижней части диалогового окна:

.DEFINE CLOCK
+ 0us 0
+ LABEL=begin
+ +5us INCR BY 1
+ +5us GOTO begin -1 TIMES

Установим параметры сигнала управления *M*. Будем исходить из предположения, что первую половину временного анализа будет занимать режим двоичного счетчика (*M* = 0); вторую половину – режим счетчика в коде Грея (*M* = 1, рисунок 4.8).

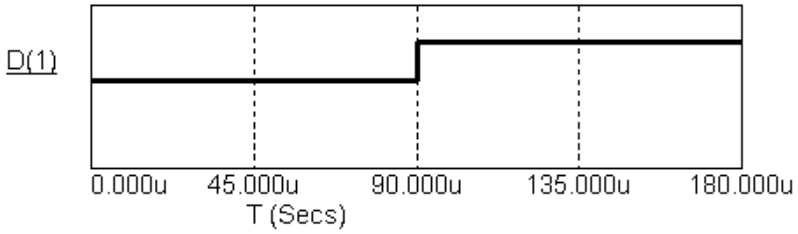


Рисунок 4.8 – Временная диаграмма сигнала управления

В диалоговом окне свойств источника *U*1 в строке FORMAT указывают значение 1. При выборе строки COMMAND задают форму сигнала управления:

.DEFINE M
+ 0us 0
+ 90us 1

Установим параметры сигнала сброса *Reset*. Сигнал сброса должен представлять собой строб-импульс, появляющийся в начальные моменты действия того или иного режима работы цифрового автомата. Учитывая инверсный тип входа CLRB, строб-импульс сброса имеет нулевой активный уровень и единичный пассивный. Длительность строб-импульса можно принять как десятую часть периода следования синхросигнала *T*/10 = 1 мкс; момент возникновения фронта строб-импульса примем как +1 мкс от начала режима работы. Цифровой автомат, имеющий два режима работы, дважды сбрасывается сигналом *Reset* в моменты $t_1 = 1$ мкс и $t_2 = \frac{\tau_{OK}}{2} + 1 = 91$ мкс (рисунок 4.9).

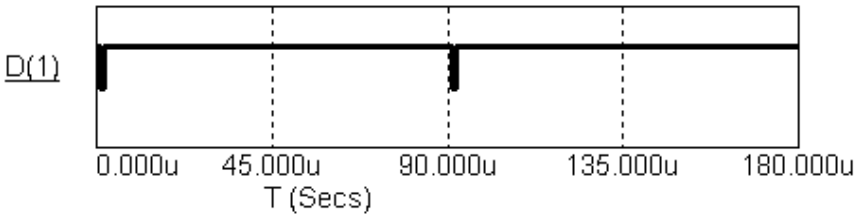


Рисунок 4.9 – Временная диаграмма сигнала сброса

В диалоговом окне свойств источника *U*7 в строке FORMAT указывают значение 1. При выборе строки COMMAND задают форму сигнала сброса:

.DEFINE RESET
+ 0us 1
+ 1us 0
+ 2us 1
+ 91us 0

+ 92us 1

В таблицу диалогового окна Transient Analysis Limits заносится следующая информация:

P	X EXPRESSION	Y EXPRESSION
1	T	D(M)
1	T	D(CLOCK)
1	T	D(RESET)
1	T	D(Q0)
1	T	D(Q1)
1	T	D(Q2)
1	T	BIN(Q2,Q1,Q0)

В последней строке таблицы записано двоичное представление трехразрядной шины данных, составленной из проводников *Q*2, *Q*1, *Q*0.

Результат исследования трехразрядного двухрежимного цифрового автомата представлен в виде временной диаграммы на рисунке 4.10. Сравнивая последовательность состояний трехразрядной шины *bin*(*Q*2,*Q*1,*Q*0) с диаграммой на рисунке 4.5, можно убедиться в адекватности проведенного исследования.

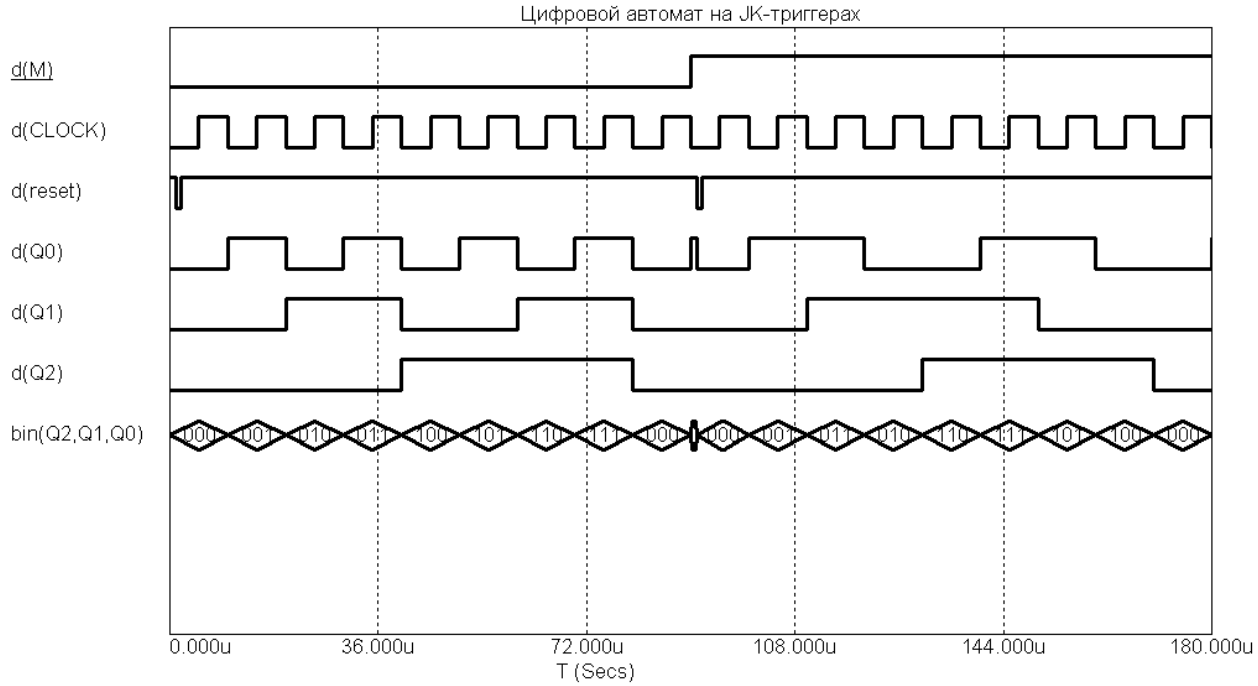


Рисунок 4.10 – Временная диаграмма работы двухрежимного трехразрядного автомата

4.5 Лабораторное задание

Требуется спроектировать трехразрядный автомат на основе *JK*-триггеров с двумя режимами работы, управляемый входным сигналом *M*. При *M* = 0 автомат должен работать как двоичный счетчик с модулем счета 8, а при *M* = 1 – согласно приведенной в варианте задания последовательности состояний. Комбинационные цепи автомата реализовать в базисе И-НЕ. Частота тактовых импульсов синхронизации 100 кГц. Правильность предложенного

схемотехнического решения подтвердить результатами моделирования в программе Micro-CAP.

4.6 Контрольные вопросы

1. Что такое последовательностные схемы?
2. В чем основное отличие автоматов с памятью от комбинационных цепей?
3. В чем заключается отличие асинхронных и синхронных автоматов с памятью?
4. Для чего служат тактирующие импульсы в синхронных автоматах с памятью?

Продолжение на следующей странице

5. Каков основной недостаток асинхронных автоматов в плане их практического использования?
6. Что такое автоматы Мили и автоматы Мура?
7. Какие существуют способы кодирования состояний автоматов?
8. На основе каких типов триггеров наиболее часто реализуют цифровые автоматы?
9. Как определить число триггеров, необходимых для синтеза цифрового автомата, если N – число состояний автомата?

4.7 Варианты заданий

№ Варианта	Последовательность состояний при $m = 1$
1	$000 \to 010 \to 011 \to 111 \to 101 \to 001 \to 110 \to 100 \to 000$
2	$000 \to 011 \to 100 \to 110 \to 001 \to 111 \to 101 \to 010 \to 000$
3	$000 \to 100 \to 101 \to 011 \to 111 \to 001 \to 110 \to 010 \to 000$
4	$000 \to 101 \to 110 \to 100 \to 001 \to 111 \to 010 \to 011 \to 000$
5	$000 \to 110 \to 111 \to 101 \to 100 \to 001 \to 010 \to 011 \to 000$
6	$000 \to 111 \to 001 \to 011 \to 010 \to 100 \to 110 \to 101 \to 000$
7	$000 \to 001 \to 010 \to 111 \to 011 \to 110 \to 100 \to 101 \to 000$
8	$000 \to 010 \to 100 \to 001 \to 111 \to 110 \to 011 \to 101 \to 000$
9	$000 \to 011 \to 101 \to 100 \to 001 \to 010 \to 111 \to 110 \to 000$
10	$000 \to 100 \to 110 \to 111 \to 101 \to 001 \to 010 \to 011 \to 000$
11	$000 \to 101 \to 111 \to 011 \to 110 \to 001 \to 010 \to 100 \to 000$
12	$000 \to 110 \to 011 \to 101 \to 111 \to 001 \to 100 \to 010 \to 000$
13	$000 \to 111 \to 010 \to 100 \to 001 \to 101 \to 011 \to 110 \to 000$
14	$000 \to 001 \to 100 \to 111 \to 110 \to 101 \to 010 \to 011 \to 000$
15	$000 \to 010 \to 101 \to 011 \to 001 \to 100 \to 111 \to 110 \to 000$
16	$000 \to 011 \to 110 \to 100 \to 101 \to 111 \to 001 \to 010 \to 000$
17	$000 \to 100 \to 111 \to 011 \to 001 \to 010 \to 101 \to 110 \to 000$
18	$000 \to 101 \to 001 \to 111 \to 010 \to 100 \to 011 \to 110 \to 000$
19	$000 \to 110 \to 011 \to 001 \to 010 \to 100 \to 101 \to 111 \to 000$
20	$000 \to 111 \to 100 \to 001 \to 010 \to 011 \to 101 \to 110 \to 000$
21	$000 \to 001 \to 111 \to 101 \to 011 \to 100 \to 110 \to 010 \to 000$
22	$000 \to 010 \to 001 \to 111 \to 101 \to 110 \to 011 \to 100 \to 000$
23	$000 \to 011 \to 001 \to 100 \to 111 \to 010 \to 110 \to 101 \to 000$
24	$000 \to 100 \to 001 \to 101 \to 011 \to 010 \to 110 \to 111 \to 000$
25	$000 \to 101 \to 100 \to 010 \to 001 \to 011 \to 110 \to 111 \to 000$

5 ЛАБОРАТОРНАЯ РАБОТА №5 – АЛЬТЕРНАТИВНЫЕ СПОСОБЫ ПРОЕКТИРОВАНИЯ АВТОМАТОВ С ПАМЯТЬЮ

5.1 Цель работы

В ходе выполнения настоящей работы предусматривается:

1) знакомство с методом синтеза перестраиваемых автоматов с памятью на основе мультиплексного управления;

2) знакомство с методом синтеза автоматов с памятью, в которых используется кодирование состояний кодами «1 из *N*»;

3) изучение работы сдвигающих регистров;

4) изучение работы шифраторов, преобразующих унитарный код в двоичные, двоично-десятичные коды, коды Грея.

5.2 Порядок выполнения работы

1. Изучить методические указания к лабораторной работе.
2. Письменно, в отчете по лабораторной работе ответить на контрольные вопросы.
3. Внимательно ознакомиться с примером, приведенным в пункте 5.5.
4. Выполнить лабораторное задание согласно варианту задания.
5. Сделать выводы по работе.

Внимание! Отчет по лабораторной работе в обязательном порядке должен содержать: схемы включения, графики зависимостей, все необходимые расчеты и их результаты, текстовые пояснения. На графиках в отчете должны присутствовать единицы измерения, масштаб, цена деления.

5.3 Автоматы с памятью на основе мультиплексного управления

Математическая модель. Рассмотрим простейший способ реализации перестраиваемого автомата с памятью. Пусть задано множество автоматных графов. Каждый из графов характеризуется числом состояний (вершин), числом входных переменных $x_1, \ldots, x_n$ и числом выходных переменных $U_1, \ldots, U_p$. Пусть максимальное число состояний среди всех графов равно R, максимальное число входных переменных равно n, максимальное число выходных переменных равно p. Построим $(m + p)$ универсальных логических модулей (УЛМ), каждый из которых реализует все функции от $(n + m)$ переменных, где $m = \log_2 R$. Соединим их так, как показано на рисунке 5.1. На этом построение перестраиваемого автомата с памятью закончено. Двойные линии на рисунке 5.1 означают, что все входы $x_1, \ldots, x_n$ и выходы $Q_1, \ldots, Q_m$ соединены с адресными входами каждого УЛМ. Сигналы $z_1, \ldots, z_k$ – это настроечные переменные, заменяемые при настройке функциями $f_1(x_1, \ldots, x_n), \ldots, f_k(x_1, \ldots, x_n)$, где k – число различных функций, используемых для настройки.

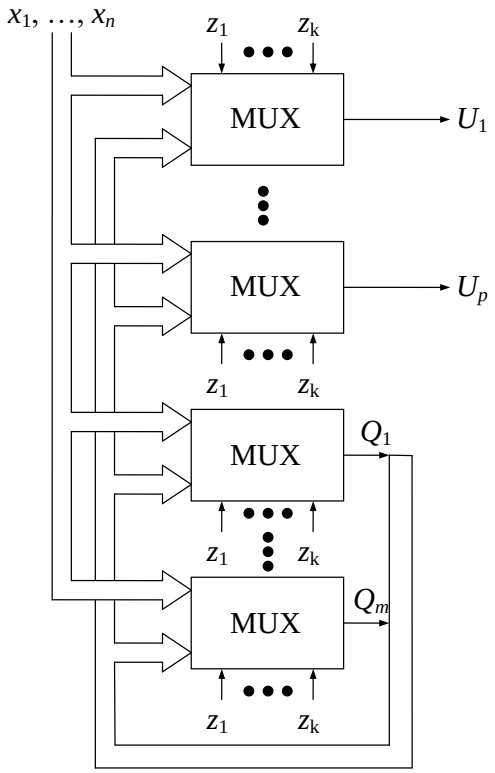


Рисунок 5.1 – Блок-схема перестраиваемого автомата с памятью
(элементы памяти условно не показаны)

Построенный перестраиваемый автомат может реализовать любой автоматный граф, имеющий не более чем R состояний, n входных переменных, p выходных переменных. Действительно, пусть задан некоторый автоматный граф, удовлетворяющий этому условию. Обозначим его входной алфавит (x_1, …, x_n), выходной алфавит (U_1, …, U_p). Закодируем вершины графа значениями дополнительных переменных (q_1, …, q_m), где $m = \log_2 R$. В качестве элементов памяти выберем D-триггеры. Тогда по графу переходов с закодированными вершинами можно построить систему функций возбуждения $\{Q\}$ и выходов $\{U\}$:

$$Q_1 = Q_1(x_1, \ldots, x_n, q_1, \ldots, q_m),$$
$$\vdots$$
$$Q_m = Q_m(x_1, \ldots, x_n, q_1, \ldots, q_m),$$
$$U_1 = U_1(x_1, \ldots, x_n, q_1, \ldots, q_m),$$
$$\vdots$$
$$U_p = U_p(x_1, \ldots, x_n, q_1, \ldots, q_m).$$

Эта система содержит $(m + p)$ функций, и каждая из них может быть реализована с помощью одного УЛМ от $(m + n)$ переменных. Перестраиваемый автомат, показанный на рисунке 5.1, состоит из $(m + p)$ УЛМ, каждый из которых реализует любую функцию от $(m + n)$ переменных. Эти УЛМ в совокупности могут реализовать любую систему из m функций переходов и p функций выходов. Таким образом, существует простой способ аппаратной реализации перестраиваемого автомата с памятью.

Аппаратная модель. Структура с мультиплексорами на входах триггеров отличается концептуальной простотой и наглядностью, для ее проектирования не требуется разработка логических преобразователей, обеспечивающих необходимые переходы автомата. Задача

решается, в сущности, табличным методом. Переменные состояния, снимаемые с триггеров, и входные сигналы образуют слово, служащее для мультиплексора адресным входом. По этому адресу в каждом мультиплексоре выбирается переменная (*0* или *1*), необходимая для перевода *D*-триггера в новое состояние. Ясно, что при этом данные для информационных входов мультиплексоров берутся прямо из таблицы переходов ($D_i = Q_{i\text{Н}}$).

Достоинство структуры – легкость перестройки автомата на новый алгоритм работы, недостаток – быстрый рост размерности мультиплексоров с ростом числа состояний и входов автомата. Структура с мультиплексорным управлением триггерами показана на рисунке 5.2. Входные сигналы x_1, …, x_n и значения разрядов слова старого состояния Q_1, …, Q_m образуют управляющее (адресное) входное слово мультиплексора, по которому выбираются значения разрядов нового слова состояния. Поступление тактового импульса вводит новое слово состояния в триггеры.

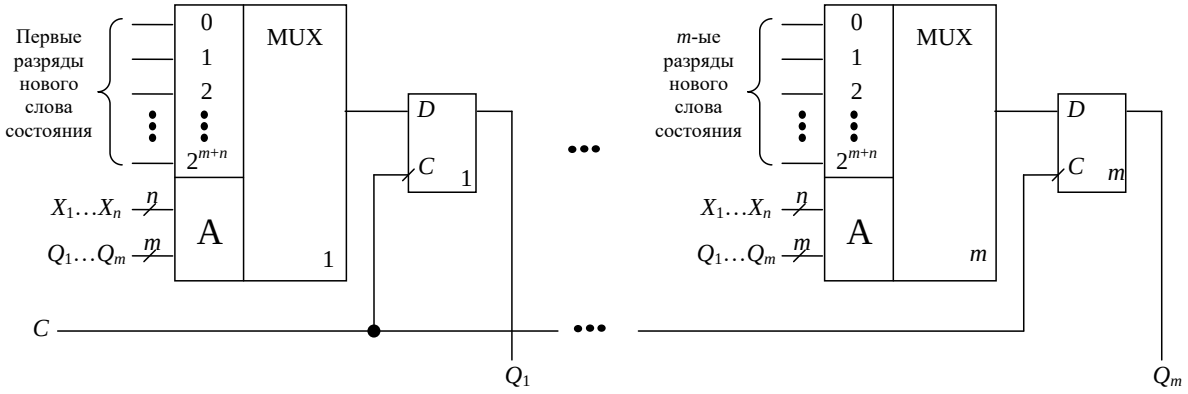


Рисунок 5.2 – Структура автомата на триггерах
с мультиплексным управлением

Рассмотрим простой пример. Пусть цифровой автомат имеет три состояния, которые можно закодировать как 00, 01, 10. Последовательность наступления таких состояний циклическая и отражена автоматным графом на рисунке 5.3.

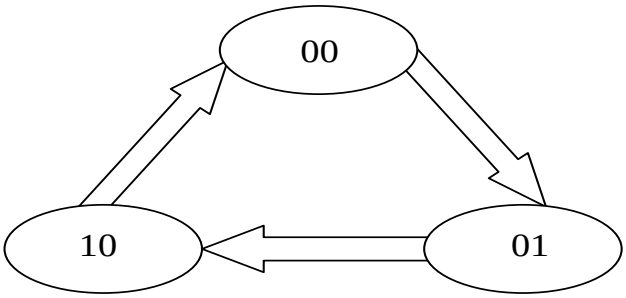


Рисунок 5.3 – Граф переходов цифрового автомата

Условимся, что младшие разряды кодовых состояний будут обозначены Q_0, старшие разряды – Q_1. Новые состояния каждого перехода будут обозначены дополнительным индексом «н»: $Q_{0\text{Н}}$, $Q_{1\text{Н}}$. Рассматривая каждый переход по отдельности (его новое и старое состояния), можно составить таблицу истинности (таблица 5.1).

Таблица 5.1 – Таблица истинности цифрового автомата

Q_1	Q_0	Q_{1H}	Q_{0H}
0	0	0	1
0	1	1	0
1	0	0	0

Таблица истинности 5.1 дает исчерпывающую информацию для синтеза цифрового автомата на основе мультиплексного управления. Количество разрядов для обозначения кодовых состояний автомата $m = 2$, значит в состав устройства должны входить два мультиплексора и два элемента памяти (рисунок 5.4). В качестве элементов памяти выбраны *D*-триггеры.

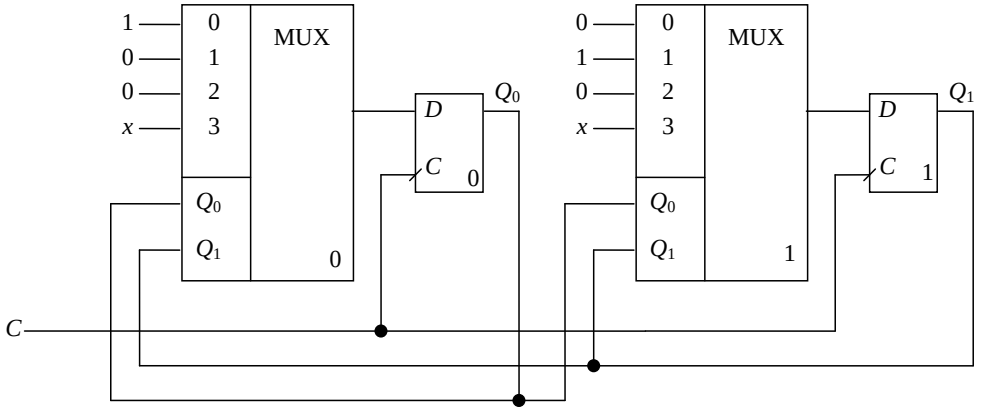


Рисунок 5.4 – Логическая схема цифрового автомата
с мультиплексным управлением

Адресная шина мультиплексоров состоит всего из двух разрядов – Q_1Q_0. Линии $X_1 \ldots X_n$, изображенные в шине адреса на рисунке 5.2, в данном случае отсутствуют, поскольку не требуется перестраивать автомат на другой алгоритм работы. Если количество адресных линий мультиплексоров $m = 2$, то количество информационных входов равно $2^m = 4$. Легко заметить, что мультиплексоры нулевого и первого разряда имеют на информационных входах столбцы констант, совпадающие с соответствующими столбцами Q_{0H} и Q_{1H} таблицы истинности. Так как граф переходов содержит только три состояния, то четвертый по счету информационный вход мультиплексоров не используется. На этих входах неопределенное (любое) значение *x*. Переход от одного состояния к другому осуществляется с помощью тактирующих импульсов *C*. Предполагается, что в начальный момент времени сигнал сброса (на схеме не показан) устанавливает *D*-триггеры в нулевое состояние, так что $Q_0 = 0$; $Q_1 = 0$.

5.4 Автоматы с памятью, имеющие кодирование состояний типа «1 из *N*»

Цифровые автоматы с памятью, имеющие структуру кодирования состояний типа «1 из *N*», состоят из трех функциональных узлов: сдвигающего регистра, шифраторов перевода исходного (1 из *N*) кода в требуемый код и мультиплексора (рисунок 5.5). Требуемый код может быть двоичным, двоично-десятичным, кодом Грея и т.д. в зависимости от поставленной задачи. В общем случае унитарный код «1 из *N*», а точнее, как следует из представленного рисунка, «1 из 2^n», может быть подвергнут 2^k различным кодовым преобразованиям. Для этой цели вводятся 2^k шифраторов перевода исходного кода в требуемый – F_1, F_2, …, F_{2^k}. По общей шине к каждому шифратору поступает исходный код «1 из 2^n» от сдвигающего регистра. Сделаем допущение о том, что, несмотря на разнообразие кодовых преобра-

зований в шифраторах, разрядность выходного кода неизменна и равна n. Выбор необходимого в текущий момент времени кода $a_n a_{n-1} \ldots a_1$ из множества доступных происходит с помощью мультиплексора. Управляющее (адресное) слово $x_k x_{k-1} \ldots x_1$ связывает выход мультиплексора с одним из информационных каналов $F_1, F_2, \ldots, F_{2^k}$ на входе.

Рассмотрим более подробно первые два функциональных узла – сдвигающий регистр и шифратор.

Параллельный (*сдвигающий*) *регистр* является, как правило, универсальным и может выполнять все доступные для регистров микрооперации. Для этого разрядные схемы, входящие в его состав, соединены между собой. На рисунке 5.6 показан цифровой автомат, который состоит из m последовательно соединенных D-триггеров, функции возбуждения которых имеют вид:

$$D_1 = x,\ D_r = Q_{r-1},\ r = 2, 3, \ldots, m. \qquad (5.1)$$

Из соотношения (5.1) вытекает, что информация, которая сохраняется в некотором такте в триггере Q_{r-1}, передается в следующем такте в триггер Q_r, т.е. происходит сдвиг информации от триггера к триггеру. Такие автоматы называются регистрами сдвига и используются для сдвига m-разрядных чисел в одном направлении. Значение входного сигнала x, которое отвечает некоторому такту, появляется на выходе регистра сдвига Q_m через m тактов.

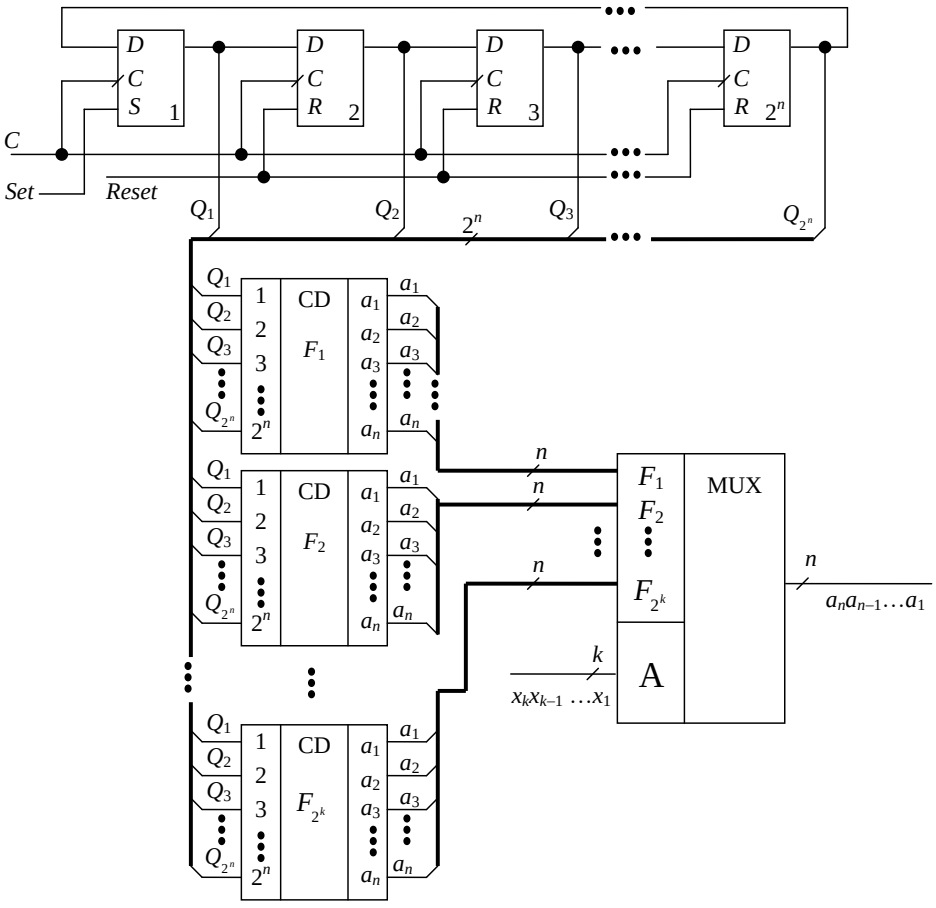


Рисунок 5.5 – Логическая схема автомата с памятью, имеющего кодирование состояний «1 из N»

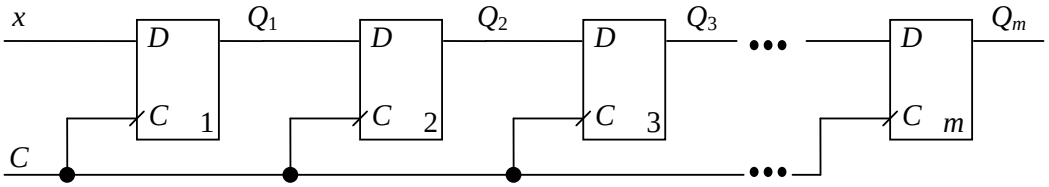


Рисунок 5.6 – Регистр сдвига

Если Q_m – старший разряд, то имеет место сдвиг в сторону старших разрядов или влево. Если же Q_m считать младшим разрядом, то будет иметь место сдвиг в сторону младших разрядов, или вправо. *Заметим, что направление сдвига в регистрах (влево или вправо) определяется не по расположению триггеров на схеме, а исходя из того, что в записи цифрового слова старшие разряды располагаются слева, а младшие справа*. Кроме основного назначения (сдвиг чисел) регистры сдвига используются и для сдвига нечисловой информации (например, при построении на них счетчиков).

Разновидностью сдвигающего регистра, применяемого в структуре кодирования «1 из *N*» (рисунок 5.5), является *кольцевой регистр*. Записанное в регистр слово содержит всего один активный уровень сигнала. При сдвигах активный уровень перемещается с одного выхода на другой, циркулируя в кольце. Число выходов кольцевого регистра равно количеству возможных состояний автомата с памятью. Недостаток кольцевого регистра – потеря правильного функционирования при сбое. Если в силу каких-либо причин слово в регистре исказится, то возникшая ошибка станет постоянной. Схема кольцевого регистра не обладает свойством автозапуска.

Специфическая ситуация может возникнуть при установке исходного состояния автомата. Если набор триггеров выполнен как единая ИС, то сброс сигналом *Reset* переведет *все* триггеры в нулевое (пассивное) состояние, тогда как первый слева (по рисунку) триггер должен получать единичное (активное) состояние. Для создания эквивалента нужной ситуации можно взять выход левого триггера с инверсного вывода, что после сброса даст на выходах регистра состояние 100…0. Чтобы не изменилось функционирование схемы, на вход левого триггера также следует подавать инвертированный сигнал (рисунок 5.7).

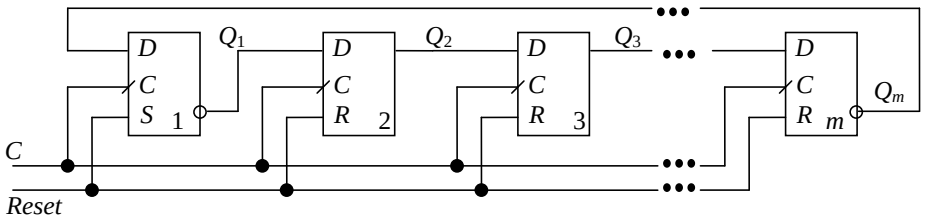


Рисунок 5.7 – Схема установки автомата в исходное состояние при использовании кода «1 из *N*»

Среди шифраторов наиболее распространены *двоичные шифраторы*, выполняющие перевод унитарного кода «1 из *N*» в двоичный код, т.е. реализуют микрооперацию, обратную микрооперации двоичных дешифраторов. Входам шифратора последовательно присваиваются значения десятичных чисел, поэтому подача активного логического сигнала на один из его входов воспринимается шифратором как подача соответствующего десятичного числа. Этот сигнал преобразуется на выходе шифратора в двоичный код. Согласно сказанному, если шифратор имеет n выходов, число его входов должно быть не более чем 2^n. Шифратор, имеющий 2^n входов и n выходов, называется *полным*. Если число входов шифратора меньше 2^n, он называется *неполным*.

Поясним работу шифратора на примере преобразователя десятичных чисел от 0 до 9 в двоично-десятичный код. Таблица истинности, соответствующая этому случаю, имеет вид представленный ниже (таблица 5.2).

Так как число входов данного устройства меньше $2^n = 16$, имеем неполный шифратор. Используя таблицу для Q_3, Q_2, Q_1, Q_0, можно записать следующие выражения:

$$\begin{aligned} Q_3 &= x_8 + x_9, \\ Q_2 &= x_4 + x_5 + x_6 + x_7, \\ Q_1 &= x_2 + x_3 + x_6 + x_7, \\ Q_0 &= x_1 + x_3 + x_5 + x_7 + x_9. \end{aligned} \qquad (5.2)$$

Полученная система характеризует работу шифратора. Логическая схема устройства, соответствующая системе (5.2), приведена на рисунке 5.8.

Таблица 5.2 – Таблица истинности шифратора

X_9	X_8	X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_0	Q_3	Q_2	Q_1	Q_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

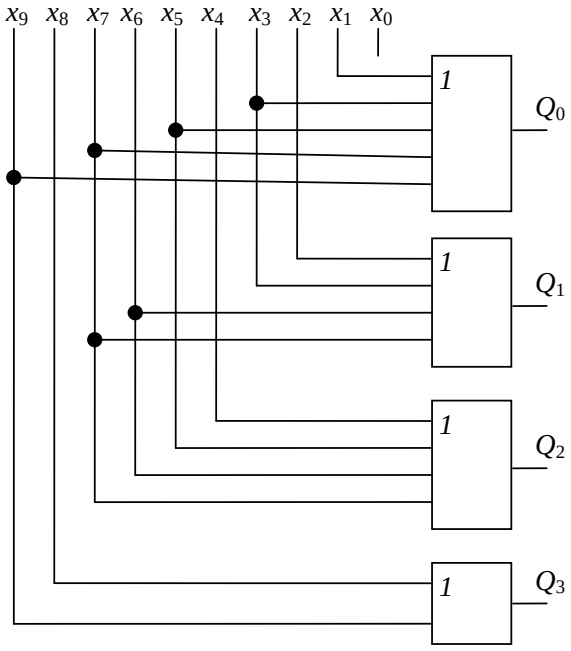


Рисунок 5.8 – Логическая схема шифратора десятичных чисел

Нетрудно заметить, что в шифраторе рассматриваемого типа сигнал, подаваемый на вход x_0, не используется. Поэтому отсутствие сигнала на любом из входов x_1, …, x_9 трактуется схемой как наличие на входе нулевого сигнала.

Структура с кодированием типа «1 из N» содержит максимальное число триггеров, т.к. в ней для каждого состояния предусматривается специальный триггер, находящийся в активном состоянии (пусть это будет состояние 1) при пассивном состоянии всех остальных. Рост числа триггеров усложняет автомат, но, одновременно с этим, резко упрощается логика,

обеспечивающая переходы автомата. Поэтому сложность автомата в целом может оказаться приемлемой.

Кодирование состояний автомата кодом «1 из *N*» (в английском языке ОНЕ, One Hot Encoding) рекомендуется для ряда современных СБИС программируемой логики, т.к. дает простой метод построения автомата высокого быстродействия, имеющего во входных цепях триггеров мало уровней логики. Метод ОНЕ устраняет много логических схем, требуемых для декодирования состояний. Набор триггеров образует структуру типа сдвигающего регистра – узла, допускающего эффективное размещение и трассировку в топологии СБИС.

5.5 Примеры проектирования трехразрядного цифрового автомата

Требуется спроектировать цифровой автомат двумя способами: на основе мультиплексного управления и на основе кодирования состояний автомата кодами «1 из *N*». Трехразрядный автомат должен иметь два режима работы (см. лабораторную работу №4). Если управляющий сигнал $M = 0$, то автомат работает как двоичный счетчик с модулем счета 8, при $M = 1$ автомат работает в коде Грея. Частота тактовых импульсов синхронизации 100 кГц. Правильность предложенных схемотехнических решений подтвердить результатами моделирования в программе MicroCAP.

I этап. Синтез цифрового автомата на основе мультиплексного управления.

При выполнении этапа исследования использованы приемы №1-7, 10 раздела «Типовые приемы работы в MicroCAP…».

Диаграмма состояний двухрежимного цифрового автомата (рисунок 5.9), а также таблица истинности (таблица 5.3) взяты нами без изменений из методического примера лабораторной работы №4.

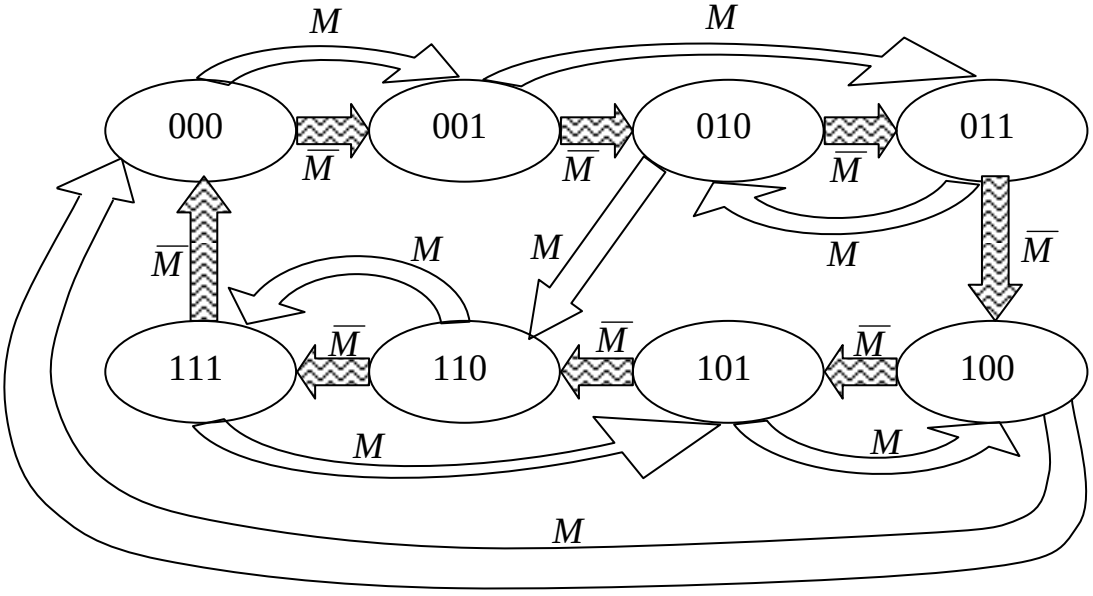


Рисунок 5.9 – Диаграмма состояний двухрежимного трехразрядного автомата

Таблица 5.3 – Таблица истинности двухрежимного цифрового автомата

Входной управляющий сигнал	Исходное состояние			Новое состояние		
M	Q_2	Q_1	Q_0	$Q_{2н}$	$Q_{1н}$	$Q_{0н}$
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	1	1
0	0	1	1	1	0	0
0	1	0	0	1	0	1
0	1	0	1	1	1	0
0	1	1	0	1	1	1
0	1	1	1	0	0	0
1	0	0	0	0	0	1
1	0	0	1	0	1	1
1	0	1	0	1	1	0
1	0	1	1	0	1	0
1	1	0	0	0	0	0
1	1	0	1	1	0	0
1	1	1	0	1	1	1
1	1	1	1	1	0	1

Конкретная реализация автомата для рассматриваемого примера (рисунок 5.10) не требует особых пояснений. Заметим лишь, что мультиплексор старшего разряда на этом рисунке расположен слева. Столбцы цифровых констант на информационных входах мультиплексоров берутся непосредственно из таблицы истинности 5.3 ($Q_{2Н}$, $Q_{1Н}$, $Q_{0Н}$). При нулевых исходных состояниях триггеров и управляющем сигнале $M = 0$ на адресные входы мультиплексоров поступает код 0000 и на входах триггеров формируется комбинация сигналов 001. Поступление тактового импульса вводит эту комбинацию в триггеры. Теперь адресом для мультиплексоров становится комбинация 0001, по которой с них снимается комбинация 010, поступающая по разрешению следующего тактового импульса в триггеры. Так реализуется режим двоичного счетчика.

Изменение управляющего сигнала M дает смену режима работы автомата. Если, например, при слове состояния 010 сигнал становится единичным, то адрес мультиплексоров изменяется с 0010 на 1010 и с их выводов снимется комбинация 110, соответствующая следующему состоянию при работе счетчика в коде Грея.

Структура и работа автомата отличаются большой наглядностью, переход к другому алгоритму функционирования требует только смены сигналов на информационных входах мультиплексоров.

Представленная на рисунке 5.10 схема цифрового автомата требует некоторой доработки в плане подготовки ее для схемотехнического моделирования в программе MicroCAP. Для этого рассмотрим логическую схему на рисунке 5.11, которая несколько отличается от исходной.

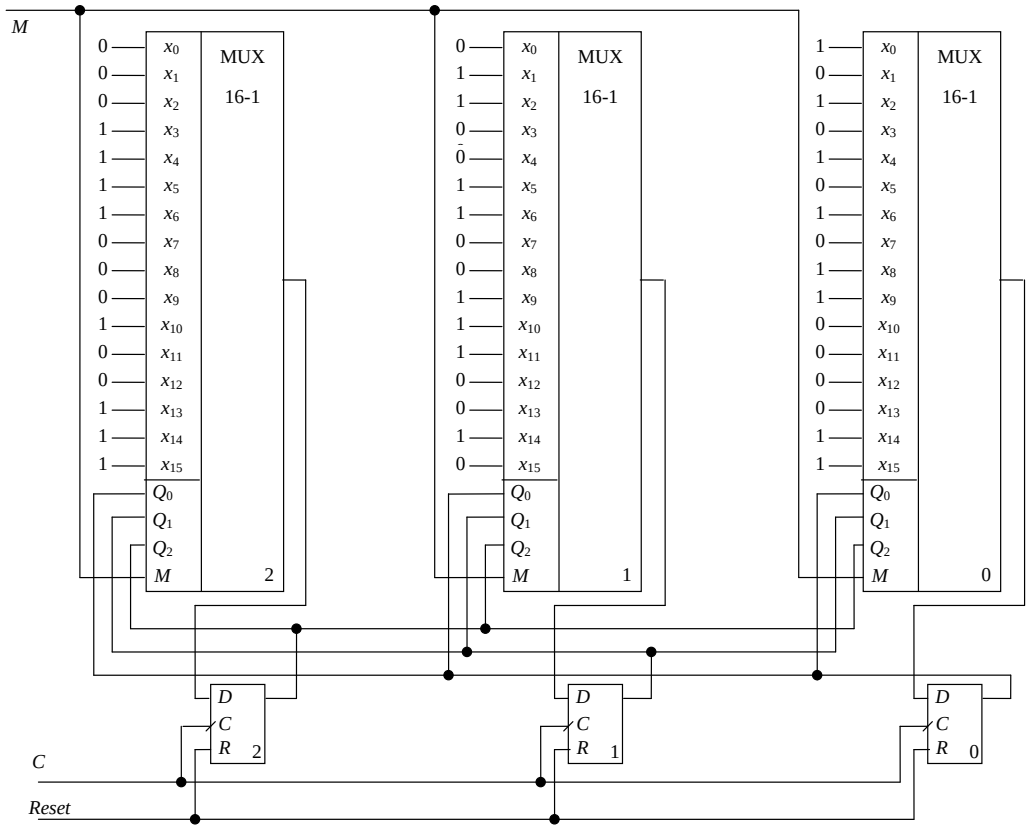


Рисунок 5.10 – Логическая схема автомата
с мультиплексорами на входах триггеров

Основой схемы являются три 16-входовых мультиплексора 74150 (отечественный аналог К155КП1), которые размещаются на поле чертежа по команде *Component/Digital Library/74xx120/148-/74150*. Устанавливать какие-либо параметры в диалоговом окне свойств мультиплексора не требуется. Условное графическое изображение мультиплексора 74150 включает следующие выводы:

- E0…E15 – информационные входы;
- A, B, C, D – адресные входы; причем вход A является младшим разрядом;
- GBAR – инверсный вход разрешения работы мультиплексора;
- W – инверсный выход мультиплексора.

Учитывая, что мультиплексоры оснащены инверсными выходами, в схему введены также инверторы *U*5, *U*7, *U*9. Элементы памяти цифрового автомата – это *D*-триггеры *U*4, *U*6, *U*8 типа DFF, рассмотренные в приеме №10

Значения цифровых констант на схеме устанавливаются с помощью элементов Fixed Digital. Чтобы не затруднять чтение и понимание схемы, применен способ скрытой прокладки проводников между выходами *D*-триггеров и адресными входами мультиплексоров.

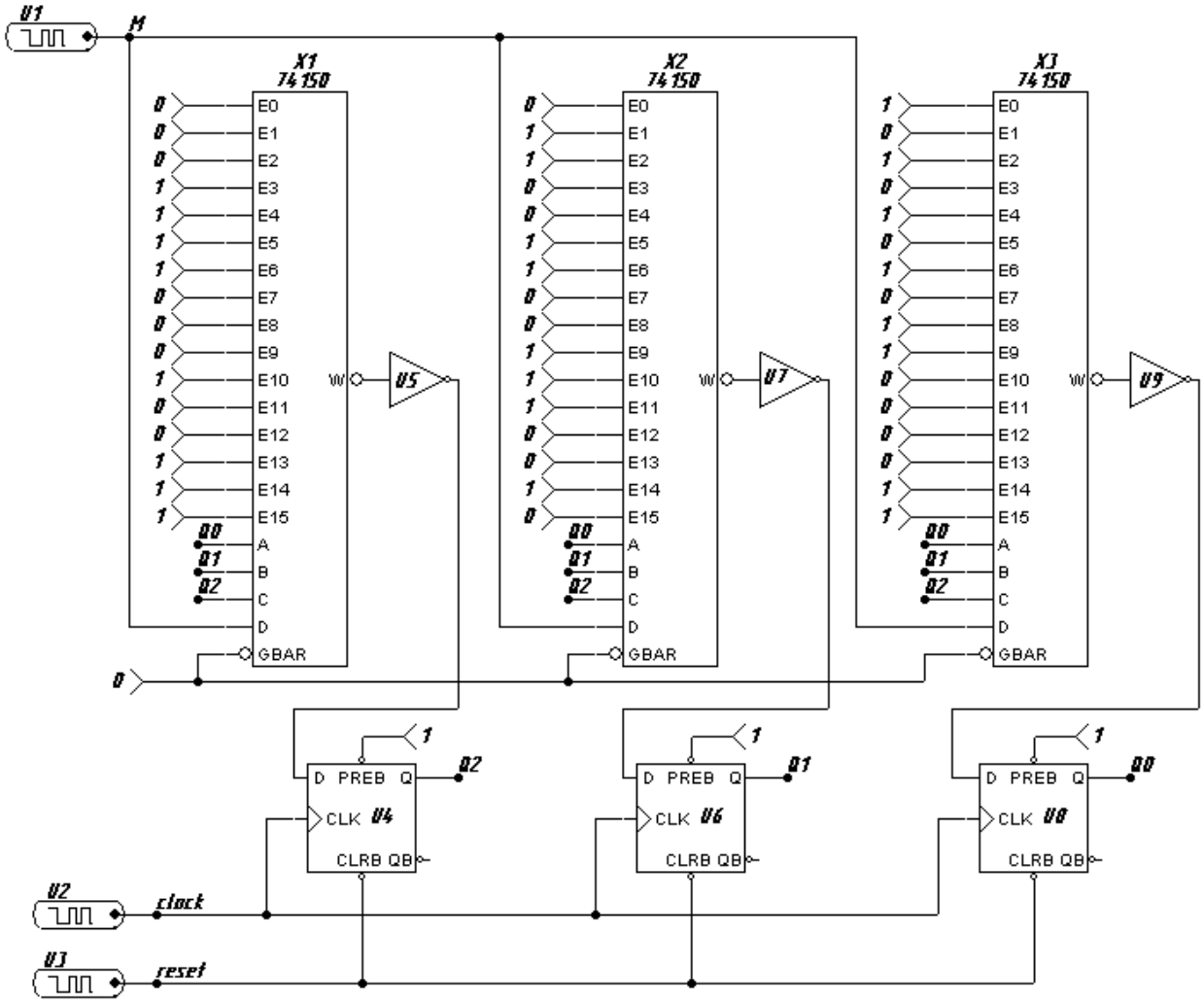


Рисунок 5.11 – Логическая схема автомата с мультиплексным управлением, подготовленная в программе MicroCAP

Применяемые в схеме *D*-триггеры активируются *передним* фронтом синхроимпульсов. Если оставить без изменений (относительно прошлой работы) параметры входных сигналов *M*, *Clock*, *Reset*, то возникнет ситуация, когда на каждом из временных интервалов в 90 мкс будут возникать вместо ожидаемых девяти 10 состояний автомата. Наиболее простой способ устранения такой ситуации – инвертирование формы синхросигнала *Clock* (рисунок 5.12). Для этого во второй строке листинга с описанием параметров синхросигнала следует изменить константу 0 на константу 1:

```
.DEFINE CLOCK
+ 0us 1
+ LABEL begin
+ +5us INCR BY 1
+ +5us GOTO begin -1 TIMES
```

Особенности установки параметров для сигналов управления *M* и сброса *Reset* подробно рассмотрены в лабораторной работе №4; в настоящей работе они остаются без изменений. Общая длительность временного вида анализа составляет 180 мкс.

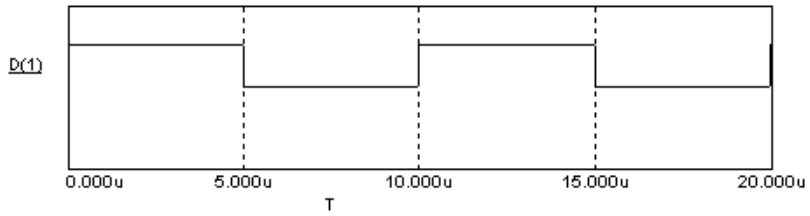


Рисунок 5.12 – Временная диаграмма синхросигнала

Моделирование схемы во временной области, а также анализ полученных результатов, не содержат принципиальных отличий от аналогичных действий в методическом примере лабораторной работы №4. Сравнивая последовательность состояний трехразрядной шины *bin*(*Q*2,*Q*1,*Q*0) на рисунке 5.13 с результатами предыдущей работы, можно убедиться в адекватности проведенного исследования.

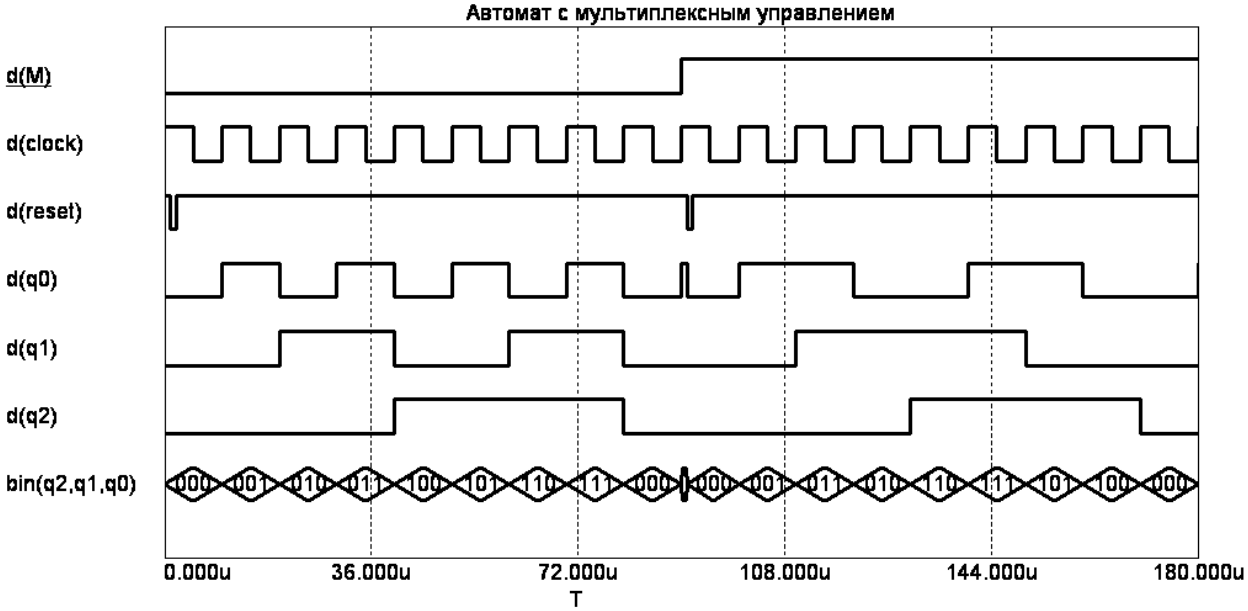


Рисунок 5.13 – Временная диаграмма работы автомата
с мультиплексным управлением

II этап. Синтез цифрового автомата на основе кодирования состояний кодами «1 из N».

При выполнении этапа исследования использованы приемы №1-6, 10 раздела «Типовые приемы работы в MicroCAP…».

Для рассматриваемого примера автомат реализуется структурой (рисунок 5.14) с числом триггеров 8. Переход в следующее состояние происходит как переход единственной единицы из триггера в соседний триггер, что осуществляется сдвигающим регистром. В структуре цифрового автомата присутствуют шифраторы для перевода кода «1 из *N*» в двоичный код, либо в код Грея, в зависимости от управляющего сигнала *M*. Этот сигнал выбирает один из двух шифраторов (они расположены в строке элементов ИЛИ); при *M* = 0 получается двоичный код, при *M* = 1 – код Грея. Автомат способен работать на высокой тактовой частоте – в цепях связи триггеров вообще нет каких-либо логических элементов.

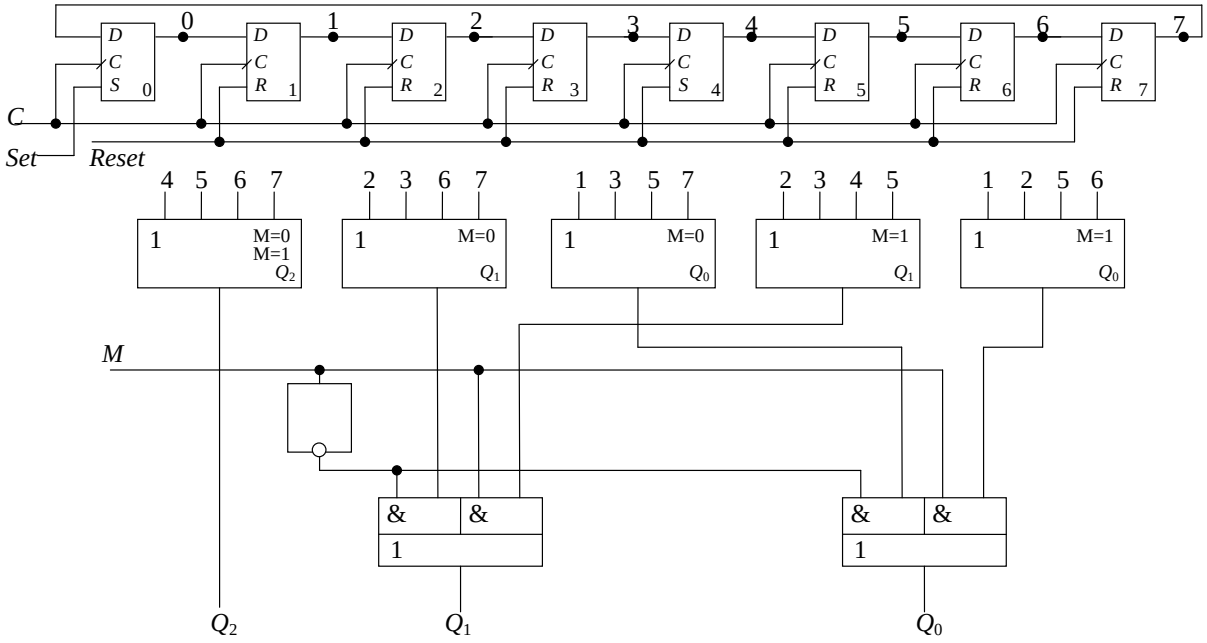


Рисунок 5.14 – Логическая схема автомата
с кодированием состояний кодами «1 из *N*»

Составим таблицы истинности, отражающие процесс кодирования: код «1 из *N*» → двоичный код; код «1 из *N*» → код Грея (таблицы 5.4, 5.5).

Предположим, что начальное состояние цифрового автомата, когда единица кода «1 из *N*» находится в крайнем левом триггере, соответствует состоянию 000 в графе переходов (рисунок 5.9) для обоих режимов функционирования. Тогда по графу переходов нужно обойти последовательно все состояния для режима $M = 0$ и для режима $M = 1$, а затем записать кодовые комбинации $Q_2Q_1Q_0$ в соответствующие строки таблиц:

$$000 \to 001 \to 010 \to 011 \to 100 \to 101 \to 110 \to 111;$$
$$000 \to 001 \to 011 \to 010 \to 110 \to 111 \to 101 \to 100.$$

Таблица 5.4 – Таблица истинности шифратора двоичного кода

X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_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	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

Таблица 5.5 – Таблица истинности шифратора кода Грея

X_7	X_6	X_5	X_4	X_3	X_2	X_1	X_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	0	0	0	1
0	0	0	0	0	1	0	0	0	1	1
0	0	0	0	1	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	1	0	1
1	0	0	0	0	0	0	0	1	0	0

Анализируя таблицы, легко заметить, что следование каноническому правилу записи логической функции в виде СДНФ приведет к неоправданно сложному выражению для разрядов Q_2, Q_1, Q_0. Поэтому в данном случае, когда в левой части таблицы $x_7 \ldots x_0$ представлен код «1 из N» для записи логических функций применяется более простое правило. Например, по таблице 5.4 находим, что в двоичном разряде Q_2 присутствует единица, когда в коде «1 из N» активными являются десятичные числа 7 ИЛИ 6 ИЛИ 5 ИЛИ 4. Переходя к выражениям булевой алгебры, имеем:

$$Q_2 = x_7 + x_6 + x_5 + x_4.$$

Полная система уравнений для случаев $M = 0$ (двоичный код) и $M = 1$ (код Грея) имеет вид:

$$M = 0 \Rightarrow \begin{cases} Q_2 = x_7 + x_6 + x_5 + x_4; \\ Q_1 = x_7 + x_6 + x_3 + x_2; \\ Q_0 = x_7 + x_5 + x_3 + x_1, \end{cases}$$

$$M = 1 \Rightarrow \begin{cases} Q_2 = x_7 + x_6 + x_5 + x_4; \\ Q_1 = x_5 + x_4 + x_3 + x_2; \\ Q_0 = x_6 + x_5 + x_2 + x_1. \end{cases}$$

Полученные системы характеризуют работу шифраторов. Выражения для Q_2 одинаковы, значит для шифрации старшего разряда служит один и тот же элемент. Шифраторы разрядов Q_1 и Q_0 представлены четырьмя отдельными логическими элементами.

Мультиплексоры реализованы в виде набора логических элементов НЕ, 2-2-И-2ИЛИ. Такой способ выбран, поскольку для переключения всего двух каналов (двоичный код/код Грея) применение функционально законченного блока мультиплексора неоправданно. В зависимости от управляющего сигнала M, цифровая константа *1* подается на левые входы либо одного, либо другого вентиля 2И каждого из элементов 2-2-И-2ИЛИ. Подача логической единицы равносильна подключению выхода соответствующего шифратора к разрядному выходу всего устройства.

Рассмотрим отличительные особенности логической схемы автомата (рисунок 5.15), подготовленной для моделирования в MicroCAP.

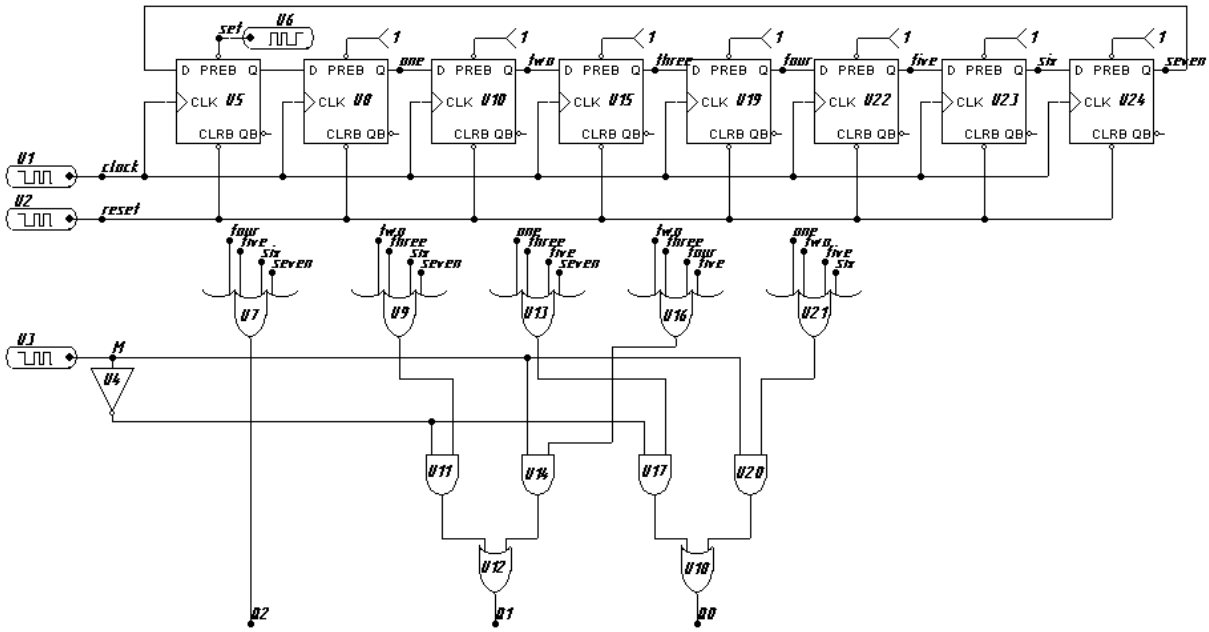


Рисунок 5.15 – Логическая схема автомата с кодированием «1 из N», подготовленная в программе MicroCAP

Помимо управляющих источников синхронизации *Clock*, сброса *Reset* и выбора режима *M*, в схеме присутствует также источник *U*6 для записи активного уровня сигнала в крайний левый триггер в начале каждого цикла. Ранее говорилось о том, что для кольцевых регистров важно в начальный момент времени записать активный уровень сигнала, который затем под воздействием синхросигналов будет непрерывно циркулировать по кольцу. В данном случае предлагается запись активного уровня реализовать в виде подачи на вход предустановки левого триггера отрицательного строб-импульса. Последовательно возникающие друг другом два строб-импульса (сброса и записи единицы) подготавливают автомат к работе в каждом цикле. Временная диаграмма таких сигналов представлена на рисунке 5.16.

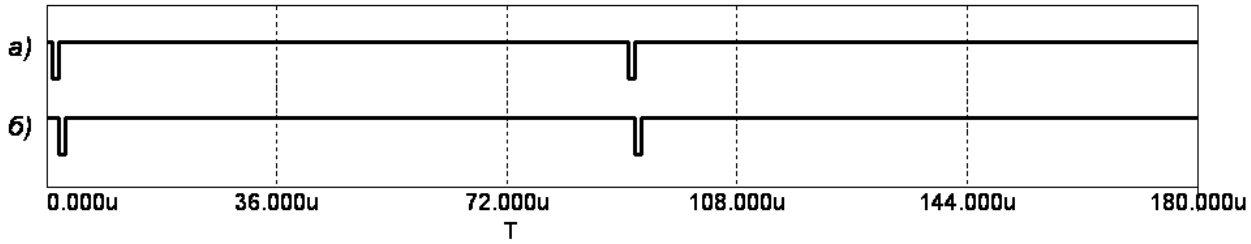


Рисунок 5.16 – Временная диаграмма сигналов сброса и записи единицы:
а) сигнал сброса; *б*) сигнал записи единицы

Ниже приведен листинг с описанием параметров сигнала записи единицы *Set* и для сравнения листинг сигнала *Reset* (из прошлой лабораторной работы). Можно заметить, что сигнал *Set* смещен во времени относительно сигнала *Reset* на 1 мкс.

```
.DEFINE SET                    .DEFINE RESET
+ 0uS 1                        + 0us 1
+ 2uS 0                        + 1us 0
+ 3us 1                  + 2us 1
+ 92us 0                       + 91us 0
+ 93us 1                       + 92us 1
```

Замечание. Схема, подготовленная для моделирования в MicroCAP (рисунок 5.15), имеет текстовое, а не числовое обозначение номеров разрядов регистра и входов шифраторов: *one*, *two*, …, *seven*. Сделано это не случайно. Следует помнить, что программа MicroCAP автоматически и на свое усмотрение проставляет номера контрольных точек на электрических схемах. По умолчанию эта информация скрыта от пользователя. Таким образом, дополнительное числовое обозначение каких-либо контрольных точек схемы приводит к конфликтной ситуации номеров, проставленных программой и пользователем. Наиболее простое решение проблемы заключается в текстовом именовании контрольных точек по мнемоническому правилу, например, написание числительных английского языка.

Моделирование схемы во временной области, а также анализ полученных результатов, не содержат принципиальных отличий от аналогичных действий на предыдущем этапе работы. Сравнивая последовательность состояний трехразрядной шины *bin*(*Q*2,*Q*1,*Q*0) на рисунке 5.17 с результатами прошлого этапа, можно убедиться в адекватности проведенного исследования.

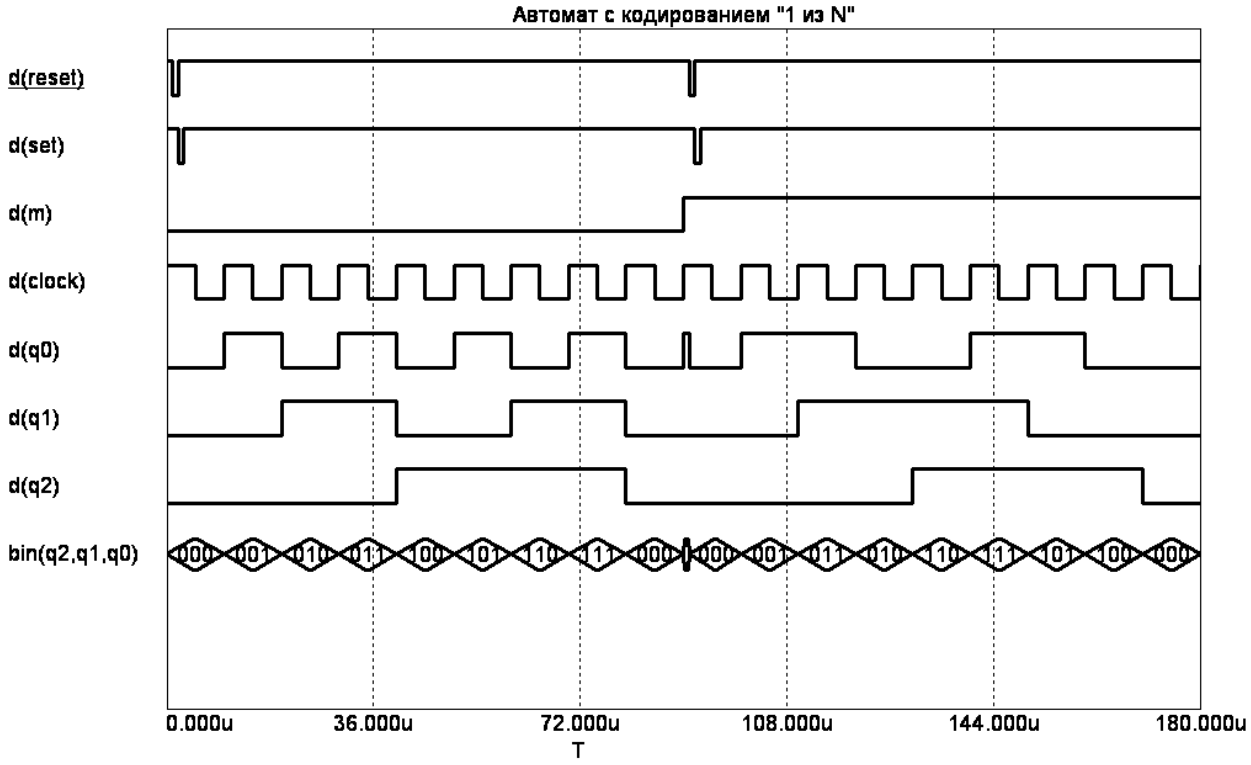


Рисунок 5.17 – Временная диаграмма работы автомата
с кодированием «1 из *N*»

5.6 Лабораторное задание

Требуется спроектировать цифровой автомат двумя способами: на основе мультиплексного управления и на основе кодирования состояний автомата кодами «1 из *N*». Трехразрядный автомат должен иметь два режима работы (см. лабораторную работу №4). Если управляющий сигнал $M = 0$, то автомат работает как двоичный счетчик с модулем счета 8, при $M = 1$ – согласно приведенной в варианте задания последовательности состояний. Частота тактовых импульсов синхронизации 100 кГц. Правильность предложенных схемотехнических решений подтвердить результатами моделирования в программе MicroCAP.

5.7 Контрольные вопросы

1. В чем заключается достоинство структуры автоматов с памятью на основе мультиплексного управления?

2. Какие сигналы образуют управляющее (адресное) слово мультиплексоров в структуре автоматов с мультиплексным управлением?

3. Для какой цели присутствуют *D*-триггеры в структуре автоматов с мультиплексным управлением?

4. Откуда берется информация для записи столбца цифровых констант на информационные входы мультиплексоров в структуре автоматов с мультиплексным управлением?

5. Какие функциональные узлы входят в состав структуры автоматов с кодированием «1 из *N*»?

6. В чем заключается основной недостаток кольцевого регистра?

7. Каким образом можно реализовать запись единицы в первый триггер кольцевого регистра, если весь набор выполнен как единая ИС?

8. Что такое полный шифратор?

9. Почему автоматы с кодированием «1 из *N*» обладают наилучшим быстродействием по сравнению с другими структурами автоматов?

5.8 Варианты заданий

№ Варианта	Последовательность состояний при $m = 1$
1	$000 \to 010 \to 011 \to 111 \to 101 \to 001 \to 110 \to 100 \to 000$
2	$000 \to 011 \to 100 \to 110 \to 001 \to 111 \to 101 \to 010 \to 000$
3	$000 \to 100 \to 101 \to 011 \to 111 \to 001 \to 110 \to 010 \to 000$
4	$000 \to 101 \to 110 \to 100 \to 001 \to 111 \to 010 \to 011 \to 000$
5	$000 \to 110 \to 111 \to 101 \to 100 \to 001 \to 010 \to 011 \to 000$
6	$000 \to 111 \to 001 \to 011 \to 010 \to 100 \to 110 \to 101 \to 000$
7	$000 \to 001 \to 010 \to 111 \to 011 \to 110 \to 100 \to 101 \to 000$
8	$000 \to 010 \to 100 \to 001 \to 111 \to 110 \to 011 \to 101 \to 000$
9	$000 \to 011 \to 101 \to 100 \to 001 \to 010 \to 111 \to 110 \to 000$
10	$000 \to 100 \to 110 \to 111 \to 101 \to 001 \to 010 \to 011 \to 000$
11	$000 \to 101 \to 111 \to 011 \to 110 \to 001 \to 010 \to 100 \to 000$
12	$000 \to 110 \to 011 \to 101 \to 111 \to 001 \to 100 \to 010 \to 000$
13	$000 \to 111 \to 010 \to 100 \to 001 \to 101 \to 011 \to 110 \to 000$
14	$000 \to 001 \to 100 \to 111 \to 110 \to 101 \to 010 \to 011 \to 000$
15	$000 \to 010 \to 101 \to 011 \to 001 \to 100 \to 111 \to 110 \to 000$
16	$000 \to 011 \to 110 \to 100 \to 101 \to 111 \to 001 \to 010 \to 000$
17	$000 \to 100 \to 111 \to 011 \to 001 \to 010 \to 101 \to 110 \to 000$
18	$000 \to 101 \to 001 \to 111 \to 010 \to 100 \to 011 \to 110 \to 000$
19	$000 \to 110 \to 011 \to 001 \to 010 \to 100 \to 101 \to 111 \to 000$
20	$000 \to 111 \to 100 \to 001 \to 010 \to 011 \to 101 \to 110 \to 000$
21	$000 \to 001 \to 111 \to 101 \to 011 \to 100 \to 110 \to 010 \to 000$
22	$000 \to 010 \to 001 \to 111 \to 101 \to 110 \to 011 \to 100 \to 000$
23	$000 \to 011 \to 001 \to 100 \to 111 \to 010 \to 110 \to 101 \to 000$
24	$000 \to 100 \to 001 \to 101 \to 011 \to 010 \to 110 \to 111 \to 000$
25	$000 \to 101 \to 100 \to 010 \to 001 \to 011 \to 110 \to 111 \to 000$

6 ЛАБОРАТОРНАЯ РАБОТА №6 – ПРОЕКТИРОВАНИЕ ДВОИЧНО-КОДИРОВАННЫХ СЧЕТЧИКОВ С ПРОИЗВОЛЬНЫМ МОДУЛЕМ

6.1 Цель работы

В ходе выполнения настоящей работы предусматривается:

1) ознакомление со способами исключения лишних состояний цифровых автоматов для построения счетчиков с произвольным модулем;

2) приобретение навыков построения счетчиков с произвольным модулем счета методом модификации межразрядных связей и методом управления сбросом;

3) изучение принципа работы параллельных счетчиков прямого и обратного счета;

4) закрепление навыков минимизации недоопределенных логических функций.

6.2 Порядок выполнения работы

1. Изучить методические указания к лабораторной работе.
2. Письменно, в отчете по лабораторной работе ответить на контрольные вопросы.
3. Внимательно ознакомиться с примером, приведенным в пункте 6.4.
4. Выполнить лабораторное задание согласно варианту задания.
5. Сделать выводы по работе.

Внимание! Отчет по лабораторной работе в обязательном порядке должен содержать: схемы включения, графики зависимостей, все необходимые расчеты и их результаты, текстовые пояснения. На графиках в отчете должны присутствовать единицы измерения, масштаб, цена деления.

6.3 Способы построения двоично-кодированных счетчиков

Счетчики с модулем, не равным целой степени числа 2, т.е. с произвольным модулем, реализуются на основе нескольких методов.

Для построения счетчика с произвольным модулем M берется разрядность $n = \lceil \log_2 M \rceil$, где $\lceil\ \rceil$ – знак округления до ближайшего справа целого числа. Иными словами, исходной структурой как бы служит двоичный счетчик с модулем 2^n, превышающим заданный и ближайший к нему. Такой двоичный счетчик имеет $2^n - M = L$ лишних (неиспользуемых) состояний, подлежащих исключению.

Способы *исключения лишних состояний* многочисленны, и для любого M можно предложить множество реализаций счетчика. Исключая некоторое число первых состояний, получим ненулевое начальное состояние счетчика, что приводит к отсутствию естественного порядка счета и регистрации в счетчике кода с избытком. Исключение последних состояний позволяет сохранить естественный порядок счета. Сложность обоих вариантов принципиально одинакова, поэтому далее будем ориентироваться на схемы с естественным порядком счета. Состояния счетчиков во всех случаях предполагаем закодированными двоичными числами, т.е. будем рассматривать двоично-кодированные счетчики.

В счетчиках с исключением последних состояний счет ведется обычным способом вплоть до достижения числа $M - 1$. Далее последовательность переходов счетчика в направлении роста регистрируемого числа должна быть прервана, и следующее состояние должно

быть нулевым. При этом счетчик будет иметь *M* внутренних состояний (от 0 до *M* – 1), т.е. его модуль равен *M*.

Остановимся на двух способах построения счетчиков с произвольным модулем: *модификации межразрядных связей* и *управлении сбросом*. При построении счетчика с модифицированными межразрядными связями последние, лишние, состояния исключаются непосредственно из таблицы функционирования счетчика. При этом после построения схемы обычным для синтеза автоматов способом получается счетчик, специфика которого состоит в нестандартных функциях возбуждения триггеров, и, следовательно, в нестандартных связях между триггерами, что и объясняет название способа. Схема получается как специализированная, изменение модуля счета требует изменение самой схемы, т.е. легкость перестройки с одного модуля на другой отсутствует. В то же время реализация схемы счетчика может оказаться простой.

При управлении сбросом выявляется момент достижения содержимым счетчика значения *M* – 1. Это является сигналом сброса счетчика в следующем такте, после чего начинается новый цикл. Этот вариант обеспечивает легкость перестройки счетчика на другие значения модуля, т.к. требуется изменять лишь код, с которым сравнивается содержимое счетчика для выявления момента сброса.

Построение счетчика первым способом. Простейшим счетчиком является счетчик по модулю 2, представляющий собой *T*-триггер при *T* = 1. Действительно, при *T* = 1 новое состояние триггера можно записать как [3]:

$$Q_H = Q\overline{T} + \overline{Q}T = \overline{Q},$$

где *Q* – старое состояние триггера; *T* – значение сигнала на информационном входе.

Для случая синхронного *T*-триггера новое состояние записывается как [3]:

$$Q_H = \left(Q\overline{T} + \overline{Q}T\right)C + \overline{C}Q = C\overline{Q} + \overline{C}Q,$$

где *C* – значение синхросигнала.

Следовательно, состояния триггера *0* и *1* циклически изменяются при каждом изменении тактового сигнала *C*. Граф переходов счетчика по модулю 2 представлен на рисунке 6.1.

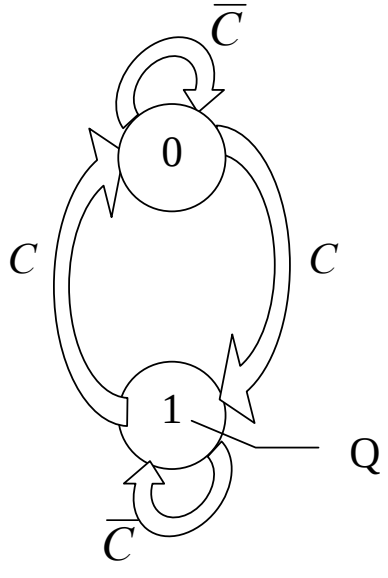


Рисунок 6.1 – Граф переходов счетчика по модулю 2

На рисунке 6.2 показан общий случай – граф переходов счетчика по модулю *M*, внутренние состояния которого обозначены числами от 0 до *M* – 1. Как уже было сказано, для получения *M* разных состояний необходимо использовать не менее $n = \lceil \log_2 M \rceil$ триггеров.

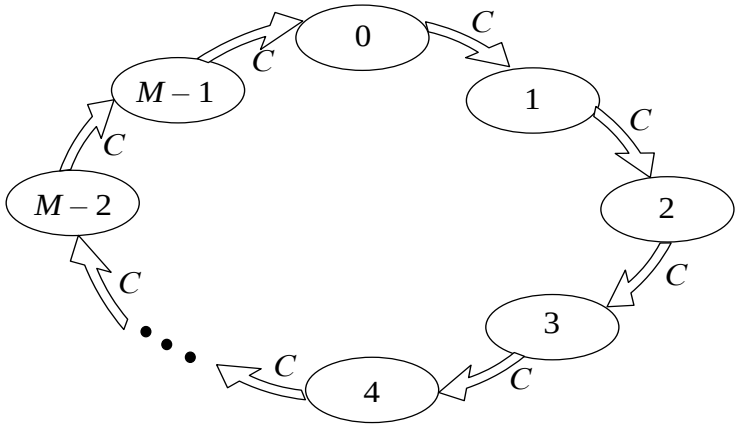


Рисунок 6.2 – Граф переходов счетчика по модулю M
(переходы при $C = 0$ не показаны)

Способ кодирования внутренних состояний счетчика может быть произвольным. Важно только, чтобы все внутренние состояния были разные. В общем случае от выбранного способа кодирования внутренних состояний автомата зависит его сложность. Закодируем внутренние состояния счетчика значениями выходных сигналов $n = \lceil \log_2 M \rceil$ триггеров Q_0, Q_1, …, Q_{n-2}, Q_{n-1}, так как это показано на рисунке 6.3.

На основании рисунка 6.3 составляется таблица истинности (таблица 6.1).

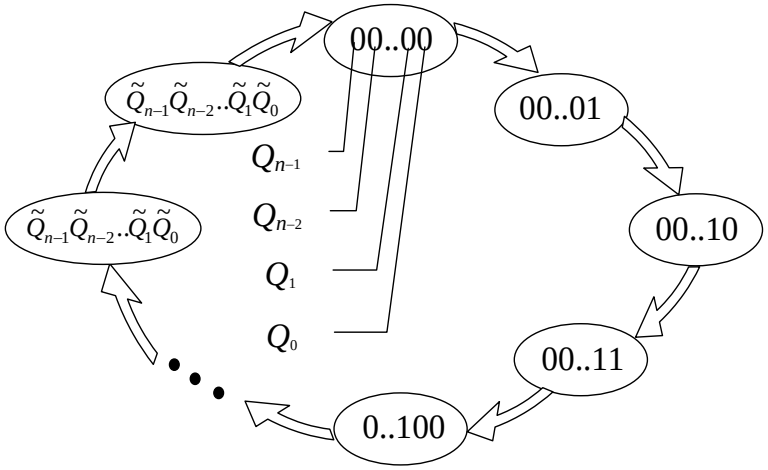


Рисунок 6.3 – Кодированный граф переходов счетчика по модулю M

Таблица 6.1 – Таблица истинности счетчика по модулю M

Q_{N-1}	Q_{N-2}	…	Q_1	Q_0	$Q_{\text{H}.N-1}$	$Q_{\text{H}.N-2}$	…	$Q_{\text{H}.1}$	$Q_{\text{H}.0}$
0	0	…	0	0	0	0	…	0	1
0	0	…	0	1	0	0	…	1	0
0	0	…	1	0	0	0	…	1	1
0	0	…	1	1	0	0	…(1)	0	0
…	…	…	…	…	…	…	…	…	…

Для синтеза счетчиков, как и любых цифровых автоматов, можно использовать триггеры любых типов: *D, T, JK*. Сложность автомата в общем случае зависит от используемых типов триггеров. Следует иметь в виду, что в одном и том же автомате можно использовать триггеры разных типов. Выполним синтез счетчика по модулю M из триггеров типа *JK*. Для этого нужно найти функции возбуждения J_i и K_i ($i = 0, 1, \ldots, n-1$). Из функции переходов *JK*-триггеров [3] вытекает, что

$$Q_{Hi} = J_i\overline{Q_i} + \overline{K}_i Q_i$$

логическое выражение с двумя неизвестными J_i и K_i, которое нужно решить относительно этих неизвестных. Пусть $Q_i = 0$, тогда $Q_{Hi} = J_i\overline{0} + \overline{K}_i 0$. Из этого уравнения следует, что $J_i = Q_{\text{H}.i}$, а $K_i = X$ – произвольное значение. Пусть теперь $Q_i = 1$, тогда $Q_{Hi} = J_i\overline{1} + \overline{K}_i 1$. Из данного уравнения следует, что $J_i = X$, а $K_i = \overline{Q}_{Hi}$. Объединив оба решения при $Q_i = 0$ и $Q_i = 1$, получим:

$$J_i = \begin{cases} Q_{H.i}, & \text{если} \quad Q_i = 0; \\ X, & \text{если} \quad Q_i = 1. \end{cases} \tag{6.1}$$

$$K_i = \begin{cases} \overline{Q}_{H.i}, & \text{если} \quad Q_i = 1; \\ X, & \text{если} \quad Q_i = 0. \end{cases} \tag{6.2}$$

Дополним исходную таблицу истинности 6.1 столбцами функций возбуждения J_i и K_i. В результате получим расширенную таблицу истинности (таблица 6.2).

Таблица 6.2 – Расширенная таблица истинности счетчика по модулю M

Q_{N-1}	Q_{N-2}	…	Q_1	Q_0	$Q_{\text{H}.N-1}$	$Q_{\text{H}.N-2}$	…	$Q_{\text{H}.1}$	$Q_{\text{H}.0}$	J_{N-1}	K_{N-1}	J_{N-2}	K_{N-2}	…	J_1	K_1	J_0	K_0
0	0	…	0	0	0	0	…	0	1	0	X	0	X	…	0	X	1	X
0	0	…	0	1	0	0	…	1	0	0	X	0	X	…	1	X	X	1
0	0	…	1	0	0	0	…	1	1	0	X	0	X	…	X	0	1	X
0	0	…	1	1	0	0	…(1)	0	0	0	X	0	X	…	X	1	X	1
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	..	…	...	…

Дополнительные столбцы функций возбуждения J_i и K_i заполняются согласно системам уравнений (6.1), (6.2). Например, для столбцов J_{n-1} и K_{n-1} имеем следующие закономерности. В четырех ячейках столбцах Q_{n-1} размещены нули, следовательно, в четыре ячейки столбца J_{n-1} нужно скопировать новые значения из столбца $Q_{\text{H}.n-1}$, т.е. нули. По этой же причине в четыре ячейки столбца K_{n-1} нужно записать произвольные значения X.

После заполнения расширенной таблицы истинности необходимо составить логические выражения для функций возбуждения J_i и K_i. Цель этого этапа – нахождение наиболее простых форм логических выражений для функций J_i и K_i, выраженных через базис старых состояний триггеров:

$$J_i = f(Q_{n-1}, Q_{n-2}, \ldots, Q_1, Q_0) \tag{6.3}$$

$$K_i = g(Q_{n-1}, Q_{n-2}, \ldots, Q_1, Q_0) \tag{6.4}$$

Очевидно, что задача нахождения минимальных форм функциональных зависимостей (6.3) и (6.4) аналогична задаче минимизации недоопределенных логических функций (см. лабораторную работу №2), поскольку значения функций J_i и K_i определены не на всех наборах аргументов.

Одно из условий минимизации недоопределенных логических функций – первоначальное ее представление в виде СДНФ. Заметим, что в общем случае для описания счетчика по модулю M в таблице истинности содержится меньшее количество строк, чем требуется для записи СДНФ:

$$M < 2^n, \tag{6.5}$$

где $n = \lceil \log_2 M \rceil$ – количество аргументов.

Учитывая неравенство (6.5), дополним таблицу истинности счетчика по модулю M количеством строк равным $2^n - M$, в каждой из которых значение логических функций J_i и K_i принимает неопределенное значение X (таблица 6.3).

Дальнейший синтез минимальных форм недоопределенных логических функций J_i и K_i ничем не отличается от последовательности, рассмотренной в лабораторной работе №2:

а) записывается СДНФ функции f_0, полученной из функции f заданием значения *0* на всех запрещенных наборах аргументов;

б) записывается СДНФ функции f_1, полученной из функции f заданием значения *1* на всех запрещенных наборах аргументов;

в) функция f_1 приводится к сокращенной форме (к форме, содержащей все простые импликанты);

г) составляется импликантная таблица из всех членов функции f_0 и простых импликант функции f_1;

д) искомая минимальная форма составляется из простых импликант функции f_1, поглощающих все члены СДНФ функции f_0.

Таблица 6.3 – Таблица истинности счетчика по модулю M с количеством строк 2^n

Q_{N-1}	Q_{N-2}	…	Q_1	Q_0	$Q_{\text{Н}.N-1}$	$Q_{\text{Н}.N-2}$	…	$Q_{\text{Н}.1}$	$Q_{\text{Н}.0}$	J_{N-1}	K_{N-1}	J_{N-2}	K_{N-2}	…	J_1	K_1	J_0	K_0
0	0	…	0	0	0	0	…	0	1	0	X	0	X	…	0	X	1	X
0	0	…	0	1	0	0	…	1	0	0	X	0	X	…	1	X	X	1
0	0	…	1	0	0	0	…	1	1	0	X	0	X	…	X	0	1	X
0	0	…	1	1	0	0	…(1)	0	0	0	X	0	X	…	X	1	X	1
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	..	…	...	…
1	1	…	1	0	X	X	…	X	X	X	X	X	X	…	X	X	X	X
1	1	…	1	1	X	X	…	X	X	X	X	X	X	…	X	X	X	X

Полученные в результате минимизации логические выражения (6.3) и (6.4) определяют аппаратную структуру комбинационных цепей на входах соответствующих триггеров (рисунок 6.4).

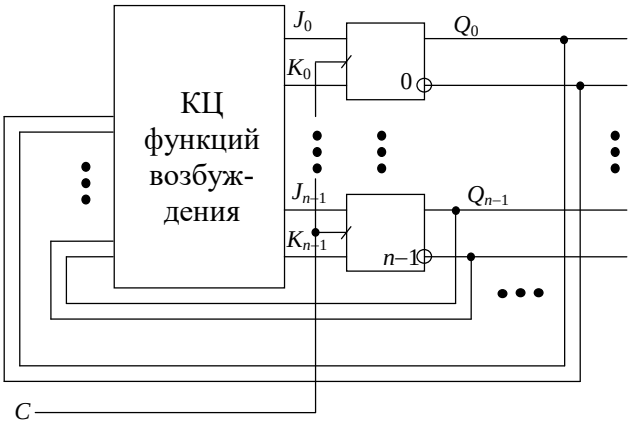


Рисунок 6.4 – Обобщенная структурная схема счетчика по модулю M

Построение счетчика вторым способом. Второй способ построения счетчиков с произвольным модулем позволяет изменить модуль счета очень простым приемом, не требующим изменений самой схемы счетчика. Сначала рассмотрим этот способ применительно к синхронному счетчику с параллельным переносом и счетом в прямом направлении (рисунок 6.5).

Функции возбуждения двоичного счетчика указанного типа, как известно [3], имеют вид $J_i = K_i = Q_0Q_1\ldots Q_{i-1}$ (причем в младшем разряде $J_0 = K_0 = 1$). Введем в эти функции сигнал сброса R, изменив их следующим образом:

$$J_i = (Q_0Q_1\ldots Q_{i-1})\overline{R},$$

$$K_i = J_i + R.$$

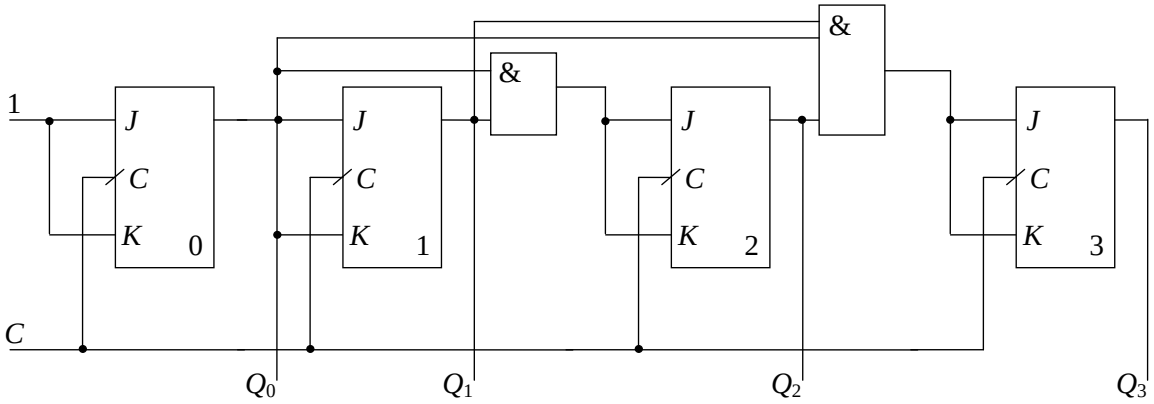
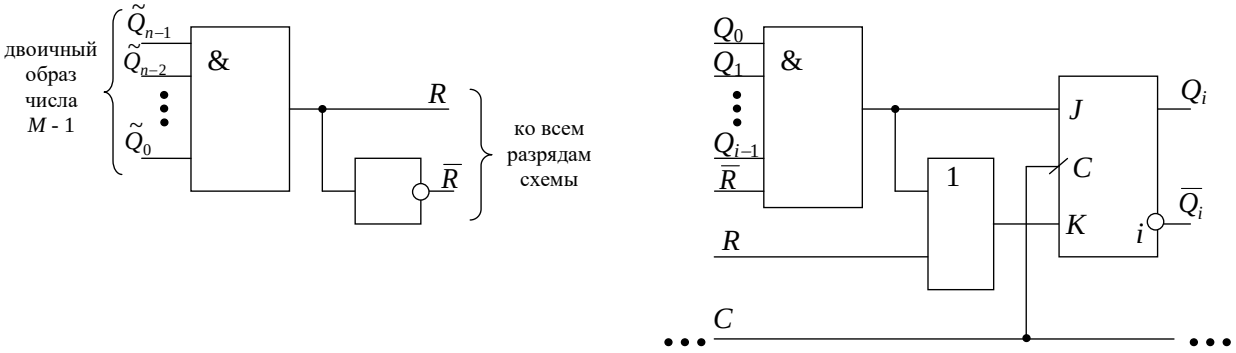


Рисунок 6.5 – Схема параллельного счетчика прямого направления

Пока сигнал сброса отсутствует ($R = 0$), функции J_i и K_i не отличаются от соответствующих функций двоичного счетчика. Когда сигнал R приобретает единичное значение, все функции J_i становятся нулевыми, K_i – единичными, что заставляет все триггеры сброситься по приходу следующего такта.

Если сигнал R появится как следствие возникновения в счетчике числа $M - 1$, то будет реализована последовательность счета 0, 1, 2, …, $M - 1$, 0, …, т.е. счетчик с модулем M.

Схемы всех разрядов счетчика с управляемым сбросом не зависят от модуля счета. Кроме разрядных схем, счетчик содержит один конъюнктор, вырабатывающий сигнал сброса при достижении содержимым счетчика значения $M - 1$ (рисунок 6.6).



а) *б*)

Рисунок 6.6 – Фрагмент схемы счетчика с управляемым сбросом:
а) схема сброса; *б*) схема *i*-ого разряда счетчика

Если, например, имеется четырехразрядный счетчик, и на входы конъюнктора выработки сигнала сброса подключены выходы триггеров, как показано на рисунке 6.7, то сброс произойдет после достижения счетчиком числа $(1001)_2 = (9)_{10}$, т.е. счетчик будет работать как двоично-десятичный.

Несколько сложнее реализуется способ управляемого сброса применительно к синхронному параллельному счетчику с обратным счетом. Известно [3], что основное отличие параллельного счетчика с обратным счетом от счетчика с прямым счетом заключается в разном подключении входов триггеров к выходам предыдущих. В частности, для схемы счетчика на *JK*-триггерах сигнал *i*-ого разряда снимается с инверсного выхода *JK*-триггера $\overline{Q_i}$ и подается затем на входы триггера следующего i+1 разряда (рисунок 6.8).

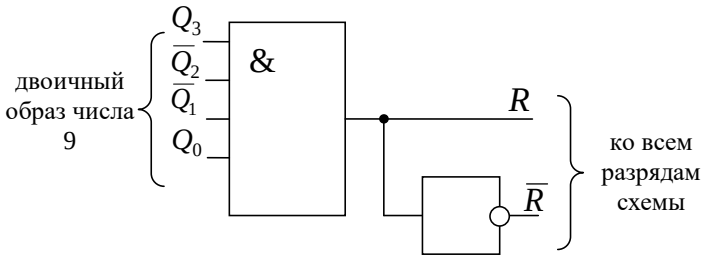


Рисунок 6.7 – Схема выработки сигнала сброса
для двоично-десятичного счетчика

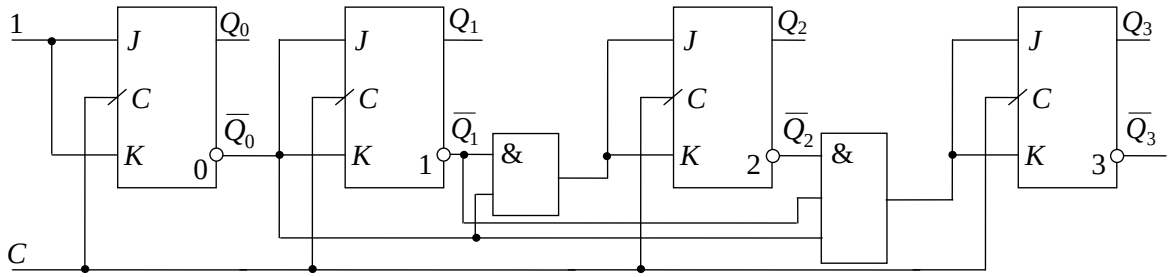


Рисунок 6.8 – Схема параллельного счетчика обратного направления

Функции возбуждения счетчика с обратным направлением счета будут иметь вид $J_i = K_i = \overline{Q}_0\overline{Q}_1...\overline{Q}_{i-1}$ (в младшем разряде по-прежнему $J_0 = K_0 = 1$). Пусть последовательность обратного счета выглядит как следующий ряд чисел:

$$M-1,\ M-2,\ \ldots,\ 1,\ 0,\ M-1,\ \ldots.$$

Тогда, в отличие от прямого счета, необходимо ввести сигнал установки *Set*. Сигнал *Set* по достижению счетчиком числа 0 устанавливает на *n* триггерах двоичный аналог числа $M-1$ для нового цикла счета (рисунок 6.9).

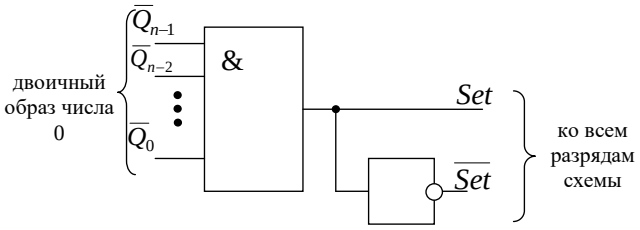


Рисунок 6.9 – Схема выработки сигнала установки

В общем случае двоичный аналог десятичного числа $M-1$ может содержать *y* единиц и *z* нулей:

$$(M-1)_{10} = \left(\tilde{Q}_{n-1}\tilde{Q}_{n-2}...\tilde{Q}_1\tilde{Q}_0\right)_2;$$

$$\sum_{i=0}^{n-1}\left(\tilde{Q}_i\Big|_{\tilde{Q}_i=1}\right) = y;\ \sum_{i=0}^{n-1}\left(\tilde{Q}_i\Big|_{\tilde{Q}_i=0}\right) = z.$$

Следовательно, разрядные схемы триггеров должны иметь две разновидности: случай установки в разряд значения *0* и случай установки значения *1*. Легко заметить, что случай установки в *i*-ый разряд триггера логического нуля аналогичен случаю выработки сигнала сброса для счетчика прямого направления:

$$J_i = \left(\overline{Q}_0\overline{Q}_1...\overline{Q}_{i-1}\right)\overline{Set}, \qquad (6.6)$$

$$K_i = J_i + Set, \qquad (6.7)$$

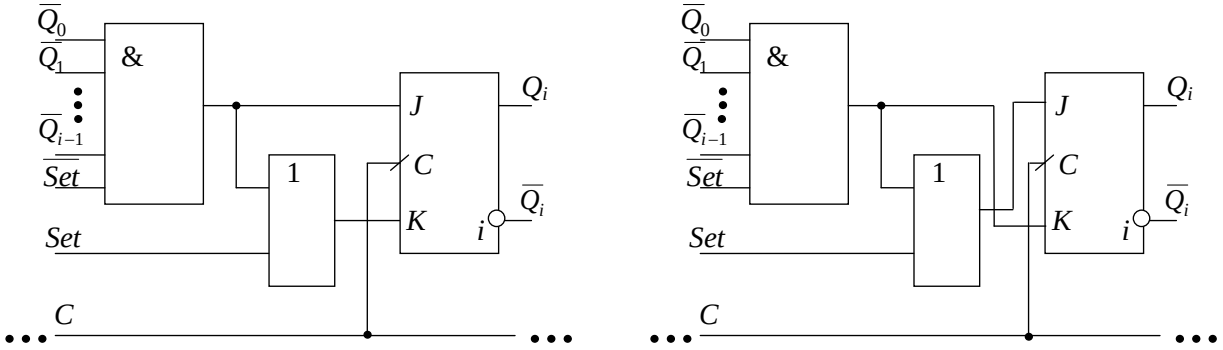
где $\overline{Q}_0, \overline{Q}_1, ..., \overline{Q}_{i-1}$ – сигналы, снимаемые с инверсных выходов *JK*-триггеров.

Для установки в разряд триггера значения логической единицы следует в выражениях (6.6), (6.7) поменять местами функции возбуждения J_i и K_i:

$$K_i = \left(\overline{Q}_0 \overline{Q}_1 ... \overline{Q}_{i-1}\right)\overline{Set},$$

$$J_i = K_i + Set.$$

Разрядные схемы триггеров для двух случаев представлены на рисунке 6.10.



а) *б*)

Рисунок 6.10 – Разрядные схемы триггеров для счетчика обратного направления:
а) установка в разряд логического нуля; *б*) установка в разряд логической единицы

6.4 Примеры проектирования двоично-кодированного счетчика с произвольным модулем

Требуется спроектировать автомат-счетчик с модулем счета $M = 15$ двумя способами: на основе модификации межразрядных связей и на основе управления сбросом. Направление счета – прямое от 0 до 14. Элементная база – триггеры типа *JK* и логические элементы И, ИЛИ, НЕ. Частота тактовых импульсов синхронизации $f = 100$ кГц. Правильность предложенных схемотехнических решений подтвердить результатами моделирования в программе MicroCAP.

I этап. Синтез счетчика на основе модификации межразрядных связей.

При выполнении этапа исследования использованы приемы №1-6, 8, 9 раздела «Типовые приемы работы в MicroCAP…».

Для построения счетчика с модулем счета $M = 15$ вычислим разрядность n цифрового автомата:

$$n = \lceil \log_2 M \rceil = \lceil \log_2 15 \rceil = \lceil 3.908 \rceil = 4.$$

Значит, исходной структурой для синтеза устройства будет служить двоичный счетчик с модулем $2^n = 16$. Такой счетчик имеет лишние состояния L, подлежащие исключению, в количестве:

$$L = 2^n - M = 16 - 15 = 1.$$

Рассмотрим последовательность заполнения таблицы истинности для счетчика (таблица 6.4). Сначала запишем заданную по условию последовательность счета в десятичной и двоичной системах счисления:

$$(0)_{10} \to (1)_{10} \to (2)_{10} \to (3)_{10} \to (4)_{10} \to (5)_{10} \to (6)_{10} \to (7)_{10} \to (8)_{10} \to (9)_{10} \to (10)_{10} \to$$
$$\to (11)_{10} \to (12)_{10} \to (13)_{10} \to (14)_{10};$$
$$(0000)_2 \to (0001)_2 \to (0010)_2 \to (0011)_2 \to (0100)_2 \to (0101)_2 \to (0110)_2 \to (0111)_2 \to$$
$$\to (1000)_2 \to (1001)_2 \to (1010)_2 \to (1011)_2 \to (1100)_2 \to (1101)_2 \to (1110)_2.$$

Таблица 6.4 – Таблица истинности счетчика со строкой лишнего состояния

Q_3	Q_2	Q_1	Q_0	$Q_{H.3}$	$Q_{H.2}$	$Q_{H.1}$	$Q_{H.0}$	J_3	K_3	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	0	0	1	0	X	0	X	0	X	1	X
0	0	0	1	0	0	1	0	0	X	0	X	1	X	X	1
0	0	1	0	0	0	1	1	0	X	0	X	X	0	1	X
0	0	1	1	0	1	0	0	0	X	1	X	X	1	X	1
0	1	0	0	0	1	0	1	0	X	X	0	0	X	1	X
0	1	0	1	0	1	1	0	0	X	X	0	1	X	X	1
0	1	1	0	0	1	1	1	0	X	X	0	X	0	1	X
0	1	1	1	1	0	0	0	1	X	X	1	X	1	X	1
1	0	0	0	1	0	0	1	X	0	0	X	0	X	1	X
1	0	0	1	1	0	1	0	X	0	0	X	1	X	X	1
1	0	1	0	1	0	1	1	X	0	0	X	X	0	1	X
1	0	1	1	1	1	0	0	X	0	1	X	X	1	X	1
1	1	0	0	1	1	0	1	X	0	X	0	0	X	1	X
1	1	0	1	1	1	1	0	X	0	X	0	1	X	X	1
1	1	1	0	0	0	0	0	X	1	X	1	X	1	0	X
1	1	1	1	X	X	X	X	X	X	X	X	X	X	X	X

Заполним сверху вниз левый блок таблицы $Q_3Q_2Q_1Q_0$ в соответствие с приведенной последовательностью двоичных состояний. Затем заполним средний блок $Q_{H.3}Q_{H.2}Q_{H.1}Q_{H.0}$, рассматривая в каждой строке левого блока $Q_3Q_2Q_1Q_0$ двоичный код как старое состояние и находя по приведенной последовательности новое состояние. Под жирной горизонтальной чертой для левого блока $Q_3Q_2Q_1Q_0$ добавим все неиспользуемые состояния. В нашем случае – это единственное состояние $(15)_{10} = (1111)_2$. Во все остальные ячейки под жирной горизонтальной чертой должны быть занесены неопределенные состояния X. Общее количество строк таблицы в итоге должно быть 2^n.

При заполнении правого блока $J_3K_3J_2K_2J_1K_1J_0K_0$ воспользуемся формулами для функций возбуждения JK-триггера (6.1) и (6.2):

$$J_i = \begin{cases} Q_{H.i}, & \text{если} \quad Q_i = 0; \\ X, & \text{если} \quad Q_i = 1, \end{cases}$$

$$K_i = \begin{cases} \overline{Q}_{H.i}, & \text{если} \quad Q_i = 1; \\ X, & \text{если} \quad Q_i = 0, \end{cases}$$

где Q_i – старое состояние i-ого триггера; $Q_{H.i}$ – новое состояние; X – неопределенное (произвольное) значение.

Поиск минимальных форм для функций возбуждения J_i и K_i аналогичен минимизации недоопределенных логических функций (см. лабораторную работу №2). Рассмотрим на примере функции возбуждения J_3 последовательность действий по поиску минимальной дизъюнктивной нормальной формы.

Доопределим столбец J_3 значениями логического нуля с последующей записью СДНФ $J_{3.0}$:

$$J_{3.0} = \overline{Q}_3Q_2Q_1Q_0.$$

Доопределим столбец J_3 значениями логической единицы с последующей записью СДНФ $J_{3.1}$. Затем, пользуясь логическим преобразователем Electronics Workbench, преобразуем СДНФ в МДНФ:

$$J_{3.1} = \overline{Q}_3Q_2Q_1Q_0 + Q_3\overline{Q}_2\overline{Q}_1\overline{Q}_0 + Q_3\overline{Q}_2\overline{Q}_1Q_0 + Q_3\overline{Q}_2Q_1\overline{Q}_0 + Q_3\overline{Q}_2Q_1Q_0 +$$
$$+Q_3Q_2\overline{Q}_1\overline{Q}_0 + Q_3Q_2\overline{Q}_1Q_0 + Q_3Q_2Q_1\overline{Q}_0 + Q_3Q_2Q_1Q_0 = Q_2Q_1Q_0 + Q_3.$$

Внимание! Для вариантов с обратным счетом возникает специфическая ситуация при использовании логического преобразователя Electronics Workbench. Напомним (см. прием №8), что наборы аргументов в таблице истинности логического преобразователя *всегда* представляют собой возрастающую последовательность двоичных чисел от $(00\ldots0)_2$ до $(11\ldots1)_2$, в то время как заполненная вручную таблица 6.4 для вариантов с обратным счетом образует иной порядок. Выход из создавшейся ситуации – переписать на бумаге таблицу 6.4, так чтобы наборы аргументов составляли возрастающую последовательность. Естественно, что при этом меняется не только набор аргументов, но и остальная информация, расположенная в строке правее.

Составим импликантную матрицу (таблица 6.5) для функции $J_{3.0}$ и для простых импликант (слагаемых) функции $J_{3.1}$. Крестиками отмечаются те столбцы членов СДНФ, которые поглощаются отдельными простыми импликантами (слагаемыми).

Таблица 6.5 – Импликантная матрица

Простые импликанты функции $j_{3.1}$	$\overline{Q}_3Q_2Q_1Q_0$
$Q_2Q_1Q_0$	×
Q_3	

Минимальная форма логического выражения функции J_3 состоит из простой импликанты, которая поглотила функцию $J_{3.0}$:

$$J_3 = Q_2Q_1Q_0\ . \qquad (6.8)$$

Аналогичным образом могут быть получены остальные минимальные формы логических функций возбуждения K_3, J_2, K_2, J_1, K_1, J_0, K_0:

$$K_3 = Q_2Q_1; \qquad (6.9)$$

$$J_2 = Q_1Q_0; \qquad (6.10)$$

$$K_2 = Q_1(Q_0 + Q_3); \qquad (6.11)$$

$$J_1 = Q_0; \qquad (6.12)$$

$$K_1 = Q_0 + Q_3Q_2; \qquad (6.13)$$

$$J_0 = \overline{Q}_3 + \overline{Q}_2 + \overline{Q}_1; \qquad (6.14)$$

$$K_0 = 1. \qquad (6.15)$$

Выражения (6.8) – (6.15) определяют структурный состав комбинационных цепей на входах *JK*-триггеров. Аппаратная реализация счетчика с модулем 15 (рисунок 6.11) сходна с реализацией цифровых автоматов с памятью.

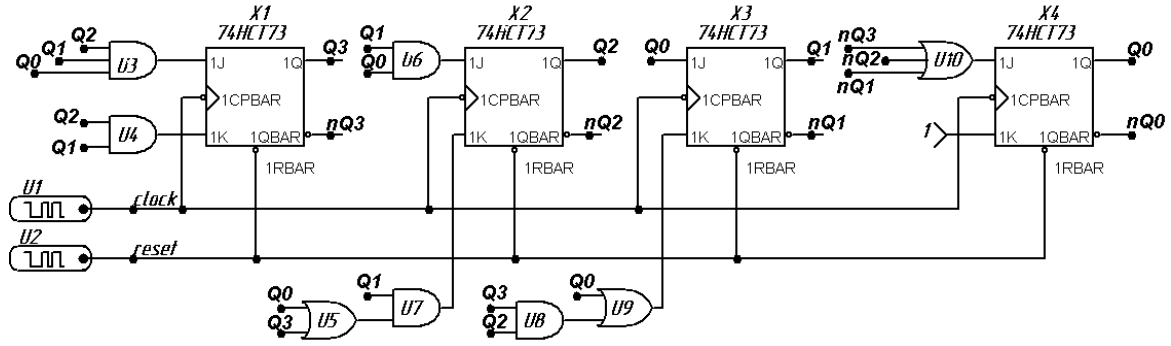


Рисунок 6.11 – Структурная схема счетчика с модулем 15
(способ модификации межразрядных связей)

В схеме применяется модель *JK*-триггера, имеющая реальный прототип – интегральную микросхему зарубежного производства 74HCT73. В корпусе микросхемы 74HCT73 со-

держится две независимые секции *JK*-триггеров, обладающие инверсным динамическим входом синхронизации. На поле чертежа каждая секция указанной микросхемы размещается по команде *Component/Digital Library/74xx42-/60-/74HCT73*.

Параметры сигнала синхронизации *Clock* можно взять без изменений из методического примера лабораторной работы №4. Сигнал сброса *Reset* должен представлять собой строб-импульс, появляющийся в начальный момент работы счетчика. Учитывая инверсный вход сброса QBAR *JK*-триггера, строб-импульс имеет нулевой активный уровень и единичный пассивный. Длительность строб-импульса примем как десятую часть периода следования синхросигнала $T/10 = 1$ мкс; момент возникновения фронта строб-импульса примем как +1 мкс от начала режима работы.

В диалоговом окне свойств источника *U*2 в строке FORMAT указывают значение 1. При выборе строки COMMAND задают форму сигнала сброса:

.DEFINE RESET
+ 0us 1
+ 1us 0
+ 2us 1

Рассчитаем длительность временного вида анализа. Частота тактовых импульсов синхронизации по условию $f = 100$ кГц, значит период следования $T = \frac{1}{f} = 10 мкс$. Количество состояний *N* счетчика равно модулю счета *M*:

$$N = M = 15.$$

При анализе в MicroCAP важно убедиться, что по достижению счетчиком максимального значения $(M - 1) = 14$ наступают новые циклы счета. По этой причине продлим временной вид анализа еще на 5 периодов следования синхроимпульсов. Тогда окончательная длительность *t* анализа составляет:

$$t = M \cdot T + 5 \cdot T = 15 \cdot 10 + 5 \cdot 10 = 200 \text{ мкс}.$$

В диалоговом окне свойств моделирования Transient Analysis Limits указывают:
- в строке ввода Time Range – длительность анализа 200u;
- в таблице – наименование функций, отображаемых на графике.

P	X EXPRESSION	Y EXPRESSION
1	T	D(CLOCK)
1	T	D(RESET)
1	T	D(Q0)
1	T	D(Q1)
1	T	D(Q2)
1	T	D(Q3)
1	T	DEC(Q3,Q2,Q1,Q0)

В последней строке таблицы с помощью ключевого слова *dec* формируется четырехразрядная шина выходного сигнала счетчика. На рисунке 6.12 представлена временная диаграмма, из которой следует, что счетчик работает в прямом направлении от 0 до 14, т.е. удовлетворяет поставленному заданию.

II этап. Синтез счетчика на основе управления сбросом.

При выполнении этапа исследования использованы приемы №1-6, 9 раздела «Типовые приемы работы в MicroCAP…».

Исходной структурой для синтеза является схема синхронного счетчика с параллельным переносом. Для обеспечения прямого направления счета необходимо, чтобы входы триггеров соединялись с прямыми выходами триггеров младших разрядов (рисунок 6.13). Количество триггеров в схеме вычисляется аналогично:

$$n = \lceil \log_2 M \rceil = \lceil \log_2 15 \rceil = \lceil 3.908 \rceil = 4.$$

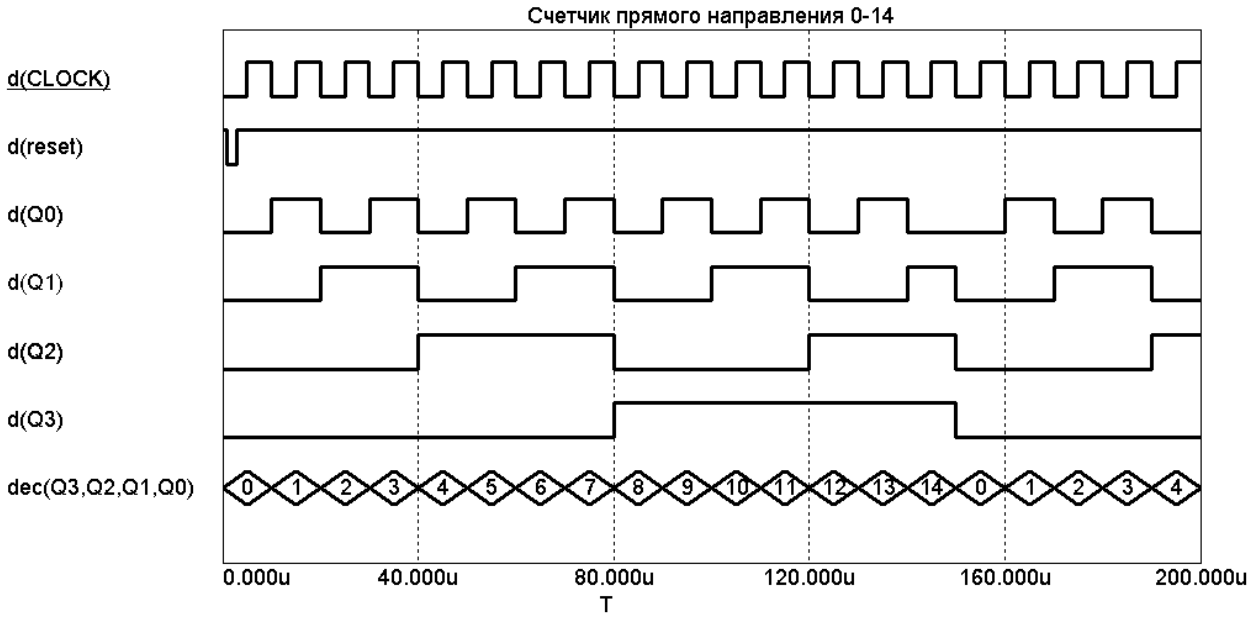


Рисунок 6.12 – Временная диаграмма работы счетчика с модулем 15
(способ модификации межразрядных связей)

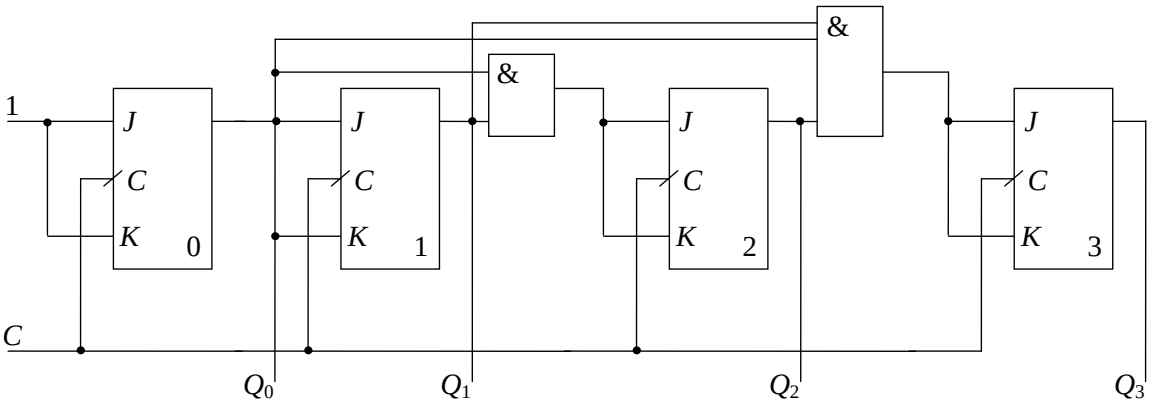


Рисунок 6.13 – Исходная схема счетчика с параллельным переносом
и модулем счета 16

Функции возбуждения счетчика по модулю 16 равны:

$$\begin{cases} J_3 = K_3 = Q_0 Q_1 Q_2; \\ J_2 = K_2 = Q_0 Q_1; \\ J_1 = K_1 = Q_0; \\ J_0 = K_0 = 1. \end{cases} \qquad (6.16)$$

Счетчик прямого направления с произвольным модулем предполагает наличие сигнала сброса R и, соответственно, схему выработки такого сигнала. Введем в исходные функции возбуждения J_i и K_i (6.16) сигнал сброса R, так чтобы:

$$J_i = \left(Q_0 Q_1 ... Q_i\right)\overline{R},$$

$$K_i = J_i + R,$$

где i = 0, 1, 2, 3.

Пока сигнал сброса отсутствует (R = 0), функции J_i и K_i не отличаются от функций счетчика (6.16) с модулем 16. Когда сигнал R приобретает единичное значение, все функции J_i становятся нулевыми, а K_i – единичными. Это заставляет все триггеры сброситься.

По условию модуль счета должен быть M = 15, значит сигнал сброса R появляется при возникновении в счетчике числа $(M-1)_{10} = (14)_{10} = (1110)_2$. Кодовая комбинация аргументов, соответствующая числу $(1110)_2$, есть $Q_3Q_2Q_1\overline{Q_0}$. Учитывая вышесказанное, схема выработки сигнала сброса может быть реализована на основе элемента 4И и элемента НЕ (рисунок 6.15). Сигнал сброса R = 1 появляется на выходе только в случае возникновения на входе схемы кода 1110.

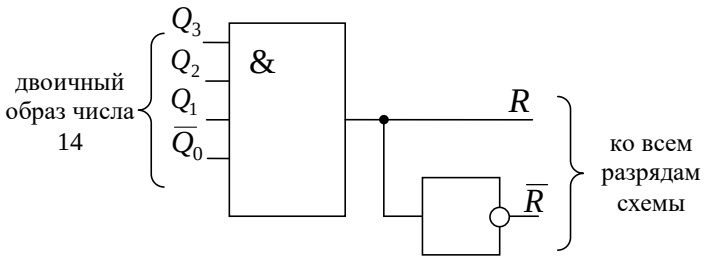


Рисунок 6.15 – Схема выработки сигнала сброса

Разрядные схемы триггеров счетчика необходимо дополнить элементом 2ИЛИ с целью учета сигнала сброса R. Кроме этого, в конъюнкторах каждого разряда следует добавить вход для подачи инверсного сигнала $\overline{R}$. Общий вид разрядной схемы триггера представлен на рисунке 6.16.

Совмещая схему выработки сигнала сброса и все разрядные схемы триггеров, получим новое схемотехническое решение – счетчик с модулем M = 15, управляемый сигналом сброса (рисунок 6.17).

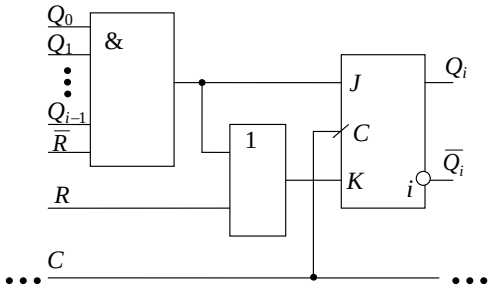


Рисунок 6.16 – Разрядная схема триггера

Заметим, что в синтезированной схеме триггеры младших разрядов расположены слева, а триггеры старших разрядов – справа. Эта особенность обусловлена структурным свойством двоичных счетчиков, содержащих триггер младшего разряда на входе схемы, т.е. слева. Технические подробности, связанные с моделированием схемы на рисунке 6.17, здесь не комментируются, поскольку аналогичны предыдущему этапу. Временные диаграммы сигналов счетчика, выполненного на основе управления сбросом, показаны на рисунке 6.18.

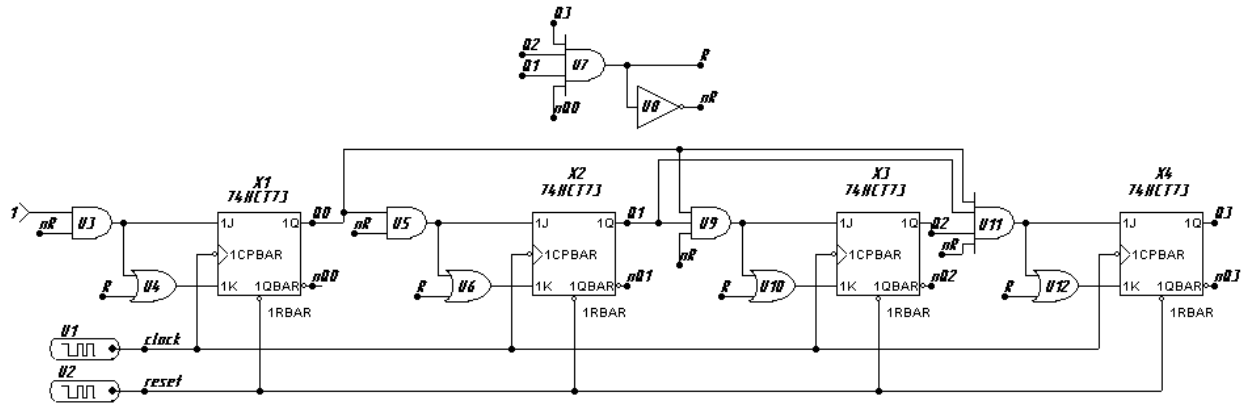


Рисунок 6.17 – Структурная схема счетчика с модулем 15
(способ управления сбросом)

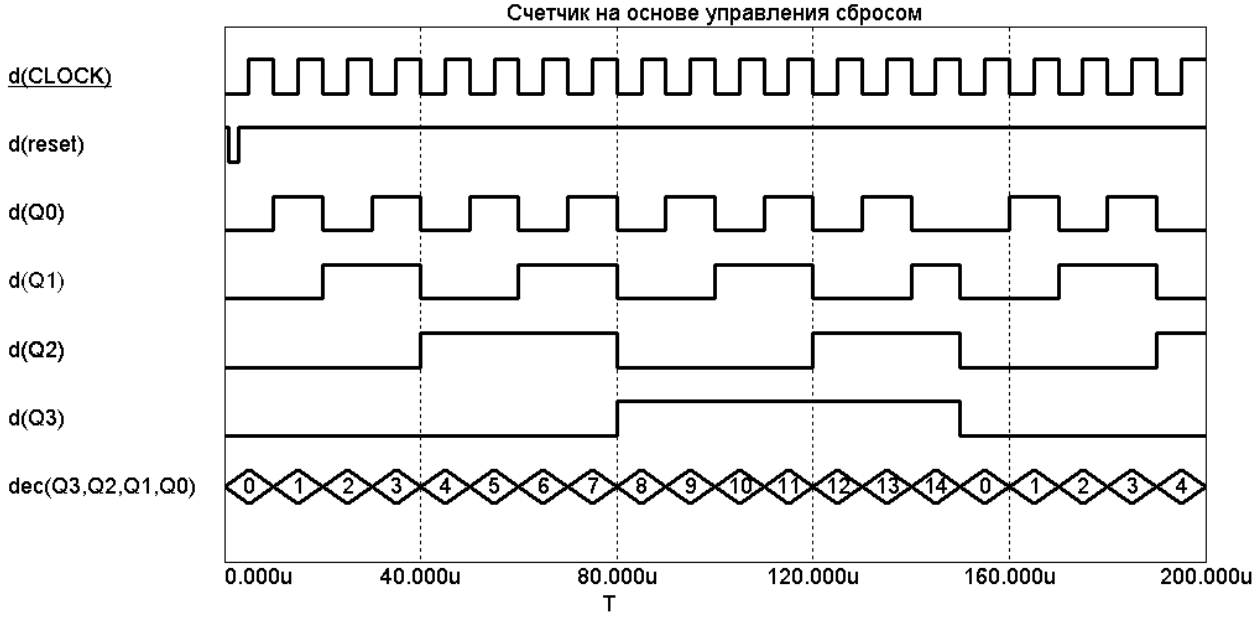


Рисунок 6.18 – Временные диаграммы работы счетчика с модулем 15
(способ управления сбросом)

Анализ временных зависимостей, их сравнение с результатами предыдущего этапа позволяет сказать об адекватности проведенного исследования.

6.5 Лабораторное задание

Варианты 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25. Спроектировать автомат-счетчик двумя способами: на основе модификации межразрядных связей и на основе управления сбросом. Направление счета – прямое, модуль счета – согласно варианту задания. Элементная база – триггеры типа *JK* и логические элементы И, ИЛИ, НЕ. Частота тактовых импульсов синхронизации f = 100 кГц. Правильность предложенных схемотехнических решений подтвердить результатами моделирования в программе MicroCAP.

Варианты 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24. Спроектировать автомат-счетчик двумя способами: на основе модификации межразрядных связей и на основе управления начальным состоянием. Направление счета – обратное, модуль счета – согласно варианту задания. *Особенности способа управления начальным состоянием для счетчиков обратного*

направления рассмотрены в конце пункта 6.3. Элементная база – триггеры типа *JK* и логические элементы И, ИЛИ, НЕ. Частота тактовых импульсов синхронизации f = 100 кГц. Правильность предложенных схемотехнических решений подтвердить результатами моделирования в программе MicroCAP.

6.6 Контрольные вопросы

1. Что такое счетчик с произвольным модулем счета *M*?
2. Как вычислить необходимое число триггеров для реализации двоично-кодированного счетчика с модулем счета *M*?
3. Какие виды триггеров можно применять для построения двоично-кодированных счетчиков?
4. Какова цель нахождения логических выражений для функций возбуждения J_i и K_i?
5. Для чего нужно дополнять таблицу истинности счетчика по модулю *M* строками лишних состояний?
6. Что такое импликантная таблица (матрица)?
7. Чему равны функции возбуждения для синхронного счетчика с параллельным переносом и счетом в прямом направлении?
8. В чем отличие реализации счетчика на основе управляемого сброса и на основе управления начальным состоянием?
9. Какие разновидности разрядных схем триггеров возможны для счетчика на основе управления начальным состоянием?

6.7 Варианты заданий

№ Варианта	Направление счета	Модуль счета	Диапазон значений
1	Прямое	5	$[0; 4]_{10}$
2	Обратное	5	$[4; 0]_{10}$
3	Прямое	6	$[0; 5]_{10}$
4	Обратное	6	$[5; 0]_{10}$
5	Прямое	7	$[0; 6]_{10}$
6	Обратное	7	$[6; 0]_{10}$
7	Прямое	9	$[0; 8]_{10}$
8	Обратное	9	$[8; 0]_{10}$
9	Прямое	10	$[0; 9]_{10}$
10	Обратное	10	$[9; 0]_{10}$
11	Прямое	11	$[0; 10]_{10}$
12	Обратное	11	$[10; 0]_{10}$
13	Прямое	12	$[0; 11]_{10}$
14	Обратное	12	$[11; 0]_{10}$
15	Прямое	13	$[0; 12]_{10}$
16	Обратное	13	$[12; 0]_{10}$
17	Прямое	14	$[0; 13]_{10}$
18	Обратное	14	$[13; 0]_{10}$
19	Прямое	17	$[0; 16]_{10}$
20	Обратное	17	$[16; 0]_{10}$
21	Прямое	18	$[0; 17]_{10}$
22	Обратное	18	$[17; 0]_{10}$
23	Прямое	19	$[0; 18]_{10}$
24	Обратное	19	$[18; 0]_{10}$
25	Прямое	20	$[0; 19]_{10}$

СПИСОК ЛИТЕРАТУРЫ

1. Разевиг, В. Д. Схемотехническое моделирование с помощью Micro-CAP 7. – Москва : Горячая линия – Телеком, 2003. – 368 с.

2. Алексеев, В. П. Системный анализ и методы научно-технического творчества: Уч. пособие / В. П. Алексеев, Д. В. Озеркин. – Томск: Издательство ИОА СО РАН, 2003. – 304 с.

3. Опадчий, Ю. Ф. Аналоговая и цифровая электроника (Полный курс): Учебник для вузов / Ю. Ф. Опадчий, О. П. Глудкин, А. И. Гуров; Под ред. О. П. Глудкина. – Москва : Горячая Линия – Телеком, 2002. – 768 с.

4. Калабеков, Б. А. Цифровые устройства и микропроцессорные системы: Учебник для техникумов связи. – Москва : Горячая линия – Телеком, 2002. – 336 с.

5. Фролкин, В.Т. Импульсные и цифровые устройства: Учеб. пособие для вузов / В. Т. Фролкин, Л. Н. Попов. – М.: Радио и связь, 1992. – 336 с.

6. Зельдин, Е. А. Цифровые интегральные микросхемы в информационно-измерительной аппаратуре. – Ленинград : Энергоатомиздат, 1986. – 280 с.

7. Потемкин, И. С. Функциональные узлы цифровой автоматики. – Москва : Энергоатомиздат, 1988. – 320 с.

8. Браммер, Ю. А. Цифровые устройства: Учебное пособие для вузов / Ю. А. Браммер, И. Н. Пащук. – Москва : Высшая школа, 2004. – 229 с.

9. Пупырев, Е. И. Перестраиваемые автоматы и микропроцессорные системы. Москва : Наука, Главная редакция физико-математической литературы, 1984. – 192 с.

10. Бойко, В. И. Схемотехника электронных систем. Цифровые устройства. – Санкт_-Петербург : БХВ-Петербург, 2004. – 512 с.

11. Micro-CAP 7.0. Electronic Circuit Analysis Program. Reference Manual – Sunnyvale: Spectrum Software, 2001. – 698 p.

12. Micro-CAP 7.0. Electronic Circuit Analysis Program. User’s Guide – Sunnyvale: Spectrum Software, 2001. – 238 p.