Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ»

И. В. ЛОМАКИН, А. И. МУХАМЕТШИН, Н. А. МАЛЁВ

АВТОМАТИЗИРОВАННОЕ ПРОЕКТИРОВАНИЯ СИСТЕМ КОНТРОЛЯ

Курсовая работа

Казань
2023

ВВЕДЕНИЕ

Увеличение производительности труда разработчиков новых изделий, сокращение сроков проектирования, повышение качества разработки проектов – важнейшие проблемы, решение которых определяет уровень ускорения научно-технического прогресса общества. Развитие систем автоматизированного проектирования (САПР) опирается на прочную научно-техническую базу. Это – современные средства вычислительной техники, новые способы представления и обработки информации, создание новых численных методов решения инженерных задач и оптимизации. Системы автоматизированного проектирования дают возможность на основе новейших достижений фундаментальных наук отрабатывать и совершенствовать методологию проектирования, стимулировать развитие математической теории проектирования сложных систем.

Применение различных программных комплексов позволяет значительно увеличить эффективность процесса проектирования. В частности, в методических указаниях рассмотрены принципы проектирования управляющих логических устройств систем автоматизации и электропривода с использованием программных пакетов *MAX+PLUS II*, *DipTrace* и *AutoCAD*.

Изучение дисциплины «Автоматизированное проектирование систем контроля» складывается из следующих последовательных этапов: 1) самостоятельного изучения теоретического материала; 2) консультации для разъяснения наиболее трудных вопросов или недостаточно рассмотренных в литературе; 3) выполнение и защита лабораторных работ; 4) выполнение и защита курсовой работы; 5) сдача экзамена.

Курсовая работа выполняется с целью закрепления полученных теоретических знаний и приобретения навыков проектирования систем и средств автоматизации с использованием вычислительной техники. Работа должна носить учебно-исследовательский характер и быть подготовительным этапом выполнения магистерской диссертации.

Выполнение курсовой работы является одной из основных форм самостоятельной работы студентов, которые в процессе про-

ектирования должны тщательно выполнить расчеты, графический материал, а также использовать современные прогрессивные, более экономичные и надежные способы для правильного решения поставленной задачи.

1. ЗАДАНИЕ К КУРСОВОЙ РАБОТЕ

1. По предложенной таблице включений построить циклограмму работы управляющего логического устройства (УЛУ) системы автоматики.

2. Используя полученную циклограмму, с учетом проверок реализации циклограммы, найти функциональное выражение для выходных параметров УЛУ.

3. Минимизировать полученные зависимости.

4. Разработать функциональную схему УЛУ на базе программируемых логических интегральных схем (ПЛИС) (аппаратная реализация), используя систему компьютерного проектирования цифровых устройств на основе *MAX+PLUS II*. Построить таблицу включений полученной системы автоматики.

5. Используя многофункциональную САПР по проектированию электронных печатных плат и созданию схемотехнической документации *DipTrace* выбрать необходимые схемные компоненты и разработать плату для проектируемого УЛУ.

6. Реализовать разрабатываемое УЛУ программным способом. Разработать алгоритм работы УЛУ, составить блок-схему. Написать и отладить программу реализации УЛУ, используя какой-либо из языков программирования. Получить таблицу включений.

7. Оформить графическую часть курсовой работы в системе автоматизированного проектирования и черчения *AutoCAD*.

2. ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ КУРСОВОЙ РАБОТЫ

Курсовая работа должна иметь титульный лист, бланк задания, подписанный руководителем, пояснительную записку и графическую часть. Пояснительная записка включает в себя оглавление, расчетно-описательную часть, выводы по всем разделам и список используемой литературы.

Пояснительная записка разбивается на разделы и подразделы. Пояснительная записка должна содержать все необходимые обоснования и расчеты с указанием принятых методов и расчетных формул. Формулы должны быть пронумерованы, буквенные обозначения расшифрованы.

В пояснительной записке приводится вывод функциональных выражений для выходных параметров управляющего логического устройства системы автоматики, показана циклограмма работы устройства, приведена принципиальная схема и проверена работоспособность логического устройства, выполненного в программной среде *MAX+PLUS II*.

Для разработки схемных компонентов, посадочных мест и создания фотошаблона печатной платы используются инструментарии пакета САПР *DipTrace*.

Кроме этого, необходимо привести алгоритм работы устройства управления и листинг программы, позволяющей обеспечить работу управляющего логического устройства с помощью промышленных микроконтроллеров, либо рабочих вычислительных станций. В качестве среды разработки программной реализации УЛУ разработки рекомендуется использовать программные пакеты *LOGO! Soft Comfort*, КОНГРАФ или *VBA MS Excel*.

Графическая часть работы выполняется в среде проектирования *AutoCAD* и содержит таблицу включений, функциональные зависимости, циклограмму работы управляющего устройства, его принципиальную схему и блок-схему алгоритма работы.

3. МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ КУРСОВОЙ РАБОТЫ

3.1. Синтез циклических автоматических систем управления

3.1.1. Основные определения

Ряд промышленных автоматических систем управления можно назвать *циклическими*. Этим термином будем называть автоматические системы, характерные тем, что у них обратная связь работает (замыкается) только периодически, через различные по продолжительности отрезки времени и охватывает в некоторых случаях не все главные механизмы, аппараты и устройства объекта управления. В горной промышленности это системы автоматизации насосных и компрессорных станций, поточно-транспортных систем (ПТС), погрузочно-разгрузочных пунктов и многое другое.

Циклические системы имеют управляющие устройства, которые осуществляют ряд логических операций, отрабатывают сложные логические функции. Системы называют циклическими потому, что после окончания определенных операций (пуска, остановки агрегата и др.) они как бы возвращаются в начальное состояние - проходят определенный цикл. Например, вода в водосборнике шахтного водоотлива достигла верхнего заданного уровня. Происходит автоматический пуск насосных агрегатов, откачка воды до заданного нижнего уровня, останов насосных агрегатов. Когда вода вновь достигнет верхнего уровня, цикл закончится. Прежде подобные системы управления в литературе часто называли дискретными, имея в виду периодическое прерывание цепи управления. Однако теперь дискретными принято называть непрерывные линейные автоматические системы

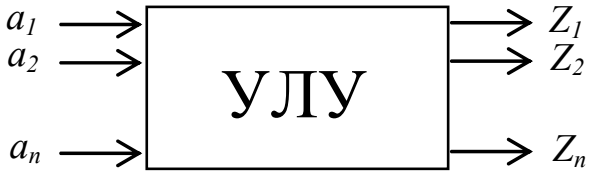

Рис.1. Управляющее логическое устройство

управления (ЛАСУ) с цифровыми регуляторами или ЭВМ. Поэтому, появилось название циклические системы, которое лучше отражает их техническую сущность. Для циклических систем управления (ЦС) используются *управляющие логические устройства* (УЛУ) (рис.1). На вход УЛУ подаются сигналы a_i от датчиков, ключей управления и др. Выходные сигналы УЛУ Z_i подаются на исполнительные механизмы. В общем виде сигнал на выходе Z_i^k в k-такте зависит не только от входных сигналов, но и от значений на выходе УЛУ в предыдущем (k-1) такте:

$$Z_i^k = f(a_1^{k-1}, a_2^{k-1}, \dots, a_n^{k-1}, Z_1^{k-1}, Z_2^{k-1}, \dots, Z_n^{k-1}) \qquad (1)$$

Существуют различные простейшие элементарные устройства, которые выполняют логические операции. Такие устройства состоят из ряда логических элементов, соединенных необходимым образом. Логические элементы могут быть реализованы на электромагнитных реле, бесконтактных магнитных реле, доменах, диодно-транзисторных схемах, мембранных или струйных пневмореле и других элементах (устройствах) автоматики. Управляющие устройства циклических систем состоят из логических элементов, соединенных в более или менее сложные схемы в соответствии с требуемый алгоритмом функционирования устройства. Кроме того, алгоритмы УЛУ могут быть реализованы не аппаратурным способом, а программным. В последнем случае в качестве управляющих устройств используются вычислительные машины (ВМ) и непосредственное цифровое управление (НЦУ). Отметим, что логические схемы на аппаратурных логических элементах («жесткая» логика) и на ВМ («гибкая» логика) могут быть построены и на базе пневматики или гидравлики, однако в настоящем методическом указании будут рассматриваться только системы с применением электрических контактных или бесконтактных логических элементов, ЭВМ, микро-ЭВМ или микропроцессоров. Предполагается, что студенты знакомы с математическим аппаратом логических схем – алгеброй логики (булевой алгеброй).

3.1.2. Математические основы построения

управляющих устройств

В алгебре логики переменные к функции – логические переменные и логические функции могут принимать только два значения, обозначаемые цифрам: 0 и 1. Двум указанным значениям ставятся в соответствие различные взаимоисключающие события (действия), условия, состояния в логических устройствах. Например, замыкание контакта – размыкание контакта, наличие сигнала – отсутствие сигнала. Важно отметить, что цифры 0 и 1 и буквенные обозначения переменных в алгебре логики, не числа, а символы и алгебра логики – это алгебра состояний.

В алгебре логики имеется множество операций. Наиболее употребительными из них является полный набор элементарных операций: например, *инверсия* «НЕ» обозначается черточкой над символом, *дизъюнкция* «ИЛИ» – знаком «+», *конъюнкция* «И» – знаком «•».

Логические элементы, которые выполняют простейшие логические операции, показаны в условных обозначениях функциональных схем по ГОСТу на рис.2.

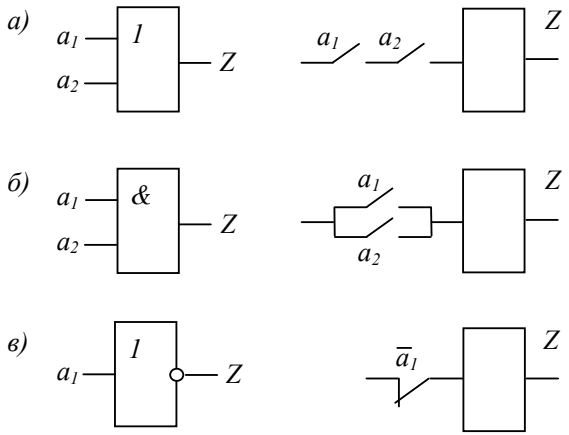

Рис.2. Функциональные схемы и релейно-контактная реализация логических элементов: а - элемент «ИЛИ», б - элемент «И», в - элемент «НЕ»

Ниже приведены основные законы булевой алгебры, позволяющие проводить тождественные преобразования формул:

1. закон нулевого множества

$$0 + a = a, 0 \cdot a = 0, 0 \cdot a \cdot b \cdot \ldots \cdot w = 0, \quad (2)$$

2. закон универсального множества

$$1 \cdot a = a, 1 + a = 1, 1 + a + b + \cdots + w = 1, \quad (3)$$

3. закон повторения

$$a \cdot a \cdot \ldots \cdot a = a, a + a + \cdots + a = a, \quad (4)$$

4. закон двойной инверсии

$$\overline{\overline{a}} = a, \quad (5)$$

5. закон дополнительности

$$a \cdot \overline{a} = 0, a + \overline{a} = 1, \quad (6)$$

6. закон коммутативности

$$a \cdot b = b \cdot a, a + b = b + a, \quad (7)$$

7. закон ассоциативности

$$\begin{gathered} a \cdot (b \cdot c) = (a \cdot b) \cdot c = a \cdot b \cdot c, \\ a + (b + c) = (a + b) + c = a + b + c, \end{gathered} \quad (8)$$

8. закон дистрибутивности

$$\begin{gathered} a \cdot (b + c) = a \cdot b + a \cdot c, \\ a + b \cdot c = (a + b) \cdot (a + c), \end{gathered} \quad (9)$$

9. закон поглощения

$$a \cdot (a + b) \cdot (a + c) \dots (a + w) = a,$$
$$a + a \cdot b + a \cdot c + \dots + a \cdot w = a, \qquad (10)$$

10. закон склеивания

$$a \cdot b + a \cdot \overline{b} = a, (a + b) \cdot \left(a + \overline{b}\right) = a, \qquad (11)$$

11. закон Моргана

$$\overline{a \cdot b \cdot c \dots \cdot w} = \overline{a} + \overline{b} + \overline{c} + \dots + \overline{w},$$
$$\overline{a + b + c + \dots + w} = \overline{a} \cdot \overline{b} \cdot \overline{c} \cdot \dots \cdot \overline{w} \dots\dots\dots\dots (12)$$

12. закон разложения

$$f(a, b, \dots, w) = a \cdot f(1, b, \dots, w) + \overline{a} \cdot f(0, b, \dots, w),$$
$$f(a, b, \dots, w) = \left(a + f(0, b, \dots, w)\right) \cdot \left(\overline{a} + f(1, b, \dots, w)\right), \qquad (13)$$

Используя законы алгебры логики, можно упростить полученное выражение. Такое упрощение необходимо выполнять всякий раз перед реализацией математического выражения «в металле».

3.1.3. Карта Карно

Карта Карно – это специального вида таблица, которая позволяет упростить процесс поиска минимальных форм и успешно применяется, когда число переменных не превосходит шести. Карты Карно для функций, зависящих от n переменных, представляет собой прямоугольник, разделенный на 2^n клеток. Каждой клетке диаграммы ставится в соответствие двоичный n-мерный набор. Значения заданной функции f из таблицы истинности вносятся в нужные квадраты. Карты Карно рассматриваются, как перестроенные соответствующим образом таблицы истинности функции, при этом их

можно рассматривать как определенную плоскую развертку n-мерного булева куба.

Карты Карно были изобретены в 1952 Эдвардом В. Вейчем и усовершенствованы в 1953 Морисом Карно, физиком из *Bell Labs*, и были призваны помочь упростить цифровые электронные схемы.

Основным методом минимизации логических функций, представленных в виде совершенной дизъюнктивной нормальной (СДНФ) или совершенной конъюнктивной нормальной (СКНФ) формах, является операция попарного неполного склеивания и элементарного поглощения. Операция попарного склеивания осуществляется между двумя термами (членами), содержащими одинаковые переменные, вхождения которых (прямые и инверсные) совпадают для всех переменных, кроме одной. В этом случае все переменные, кроме одной, можно вынести за скобки, а оставшиеся в скобках прямое и инверсное вхождение одной переменной подвергнуть склейке.

Таким образом, главной задачей при минимизации СДНФ и СКНФ является поиск термов, пригодных к склейке с последующим поглощением, что для больших форм может оказаться достаточно сложной задачей. Карты Карно предоставляют наглядный способ отыскания таких термов.

Карта Карно может быть составлена для любого количества переменных, однако удобно работать при количестве переменных не более шести. По сути, карта Карно – это таблица истинности, составленная в 2-мерном виде. На пересечении строки и столбца проставляется соответствующее значение из таблицы истинности. После того как Карта заполнена, можно приступать к минимизации.

Если необходимо получить минимальную дизъюнктивную нормальную форму (ДНФ), то в карте рассматриваем только те клетки, которые содержат единицы, если нужна конъюнктивная нормальная форма (КНФ), то рассматриваем те клетки, которые содержат нули. Минимизация производится по следующим правилам (на примере ДНФ):

1. Объединяем смежные клетки, содержащие единицы, в контур так, чтобы один контур содержал 2^n клеток (помним про то, что крайние

строки и столбцы являются соседними между собой), в контуре не должно находиться клеток, содержащих нули;
2. Контур должен располагаться симметрично оси(ей);
3. Несмежные контуры, расположенные симметрично оси(ей), могут объединяться в один;
4. Контур должен быть как можно больше, а их количество - как можно меньше;
5. Контуры могут накладываться друг на друга.

Далее, рассматриваем первый контур и определяем переменные, которые не меняются в пределах этого контура, выписываем конъюнкцию этих переменных; если неменяющаяся переменная нулевая, проставляем над ней инверсию. Данная операция выполняется для всех единичных контуров, и в итоге, полученные конъюнкции контуров объединяются дизъюнкцией.

Для КНФ выполняется всё аналогичным образом, только рассматриваются клетки с нулями, неменяющиеся переменные в пределах одной области объединяются в дизъюнкции (инверсии проставляются над единичными переменными), а полученные дизъюнкции контуров объединяются в конъюнкцию. На этом минимизация считается законченной.

В качестве примера рассмотрим логическую функцию f трех переменных, заданную следующей таблицей истинности:

Таблица 1

a	1	1	1	1	0	0	0	0
b	1	1	0	0	1	1	0	0
c	1	0	1	0	1	0	1	0
f	1	0	1	1	0	0	1	1

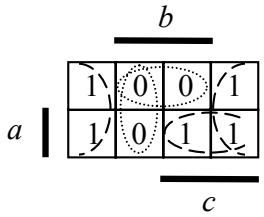

Рис.3. Карта Карно логической функции трех переменных

Карта Карно для этой функции представлена на рис.3. Построив нулевые и единичные контуры, получим следующие минимизированные выражения для f, записанные в дизъюнктивной и конъюнктивной формах:

$$f_{\text{ДНФ}} = \overline{b} + a \cdot c,$$
$$f_{\text{КНФ}} = (\overline{b} + c) \cdot (\overline{b} + a), \quad (14)$$

3.1.4. Метод циклограмм

Метод структурного синтеза на основе циклограмм является развитием известного в инженерной практике метода синтеза на основе таблиц включения. При проектировании устройств управления механизмами, работающими циклически, метод записи условий их работы посредством циклограмм имеет преимущество наглядности, а поэтому рекомендуется к использованию.

Циклограмма – это графическое изображение последовательности работы отдельных элементов управляющего логического устройства во времени. Работа элементов дискретного действия в логическом устройстве характеризуется появлением и исчезновением сигналов в определенной последовательности.

Наличие сигнала изображается на циклограмме отрезком горизонтальной прямой. Толстой линией обозначаются сигналы входных и выходных элементов, тонкой – дополнительных промежуточных элементов, пунктиром – условное включение элемента. Условное включение элемента соответствует условному состоянию, т.е. включенное или выключенное состояние этого элемента в данный период не оказывает влияние на состояние выходных элементов устройства. Слева от отрезка, отражающего работу элемента, на границе циклограммы проставляется обозначение соответствующего сигнала. Последовательность работы элементов определяется положением концов отрезков, изображающих их работу, относительно левой границы циклограммы. На циклограмме отображается любое изменение состояния элементов и указывается собственное время их срабатывания. Воздействие одного элемента на другой изображается на циклограмме стрелкой, указывающей направление воздействия (рис.4). В циклограмме время не оценивается количественно, поэтому они выполняется без масштаба. Отмечаются лишь факт срабатывания элемента,

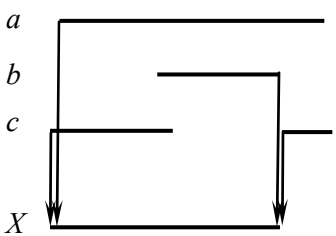

Рис.4. Пример циклограммы работы УЛУ

факт наличия или отсутствия сигнала. При наличии специального элемента задержки его сигнал на циклограмме обозначается буквой *T,* а время, по истечении которого он проявляется или исчезает, – буквой *t*. *Тактами* называются периоды, в течение которых в схеме не изменяется состояние не одного из входных, промежуточных или выходных сигналов. Каждое изменение состояния одного или одновременно нескольких элементов является началом нового такта.

Периодом включения элемента называется непрерывный ряд тактов, в течение которого этот элемент находится во включенном состоянии. *Периодом отключения* элемента называется непрерывный ряд тактов, в течение которого этот элемент находится в отключенном состоянии. Период включения элемента обозначается чертой. В периоде отключения черта на циклограмме отсутствует.

Включающим тактом называется такт, предшествующий периоду включения данного элемента. Отключающим тактом называется такт, предшествующий периоду Отключения данного элемента. Включающий период состоит из включающего такта и периода включения без отключающего такта. Отключающий период состоит из отключающего такта и периода отключения без включающего такта (понятие отключающего периода вводится при наличии нескольких периодов включения). Элементы, изменяющие свое состояние во включающем и отключающем тактах рассматриваемого периода включения, называется основными элементами.

Последовательность применения метода циклограмм предусматривает вначале построение циклограмм по словесному описанию – технологическому заданию, а затем составление по выполненной циклограмме алгебраического выражения – алгоритма функционирования УЛУ. Функциональные выражения составляются для каждого из выходных параметров для каждого периода включения и далее объединяются.

Основная формула метода циклограмм имеет вид:

$$X_i = f'_{(x_i)} \cdot \overline{f''_{(x_i)}} \qquad (15)$$

где $f'_{(x_i)}$, $f''_{(x_i)}$ - включающее и отключающее события для выходного элемента X_i в *i*-периоде включения.

Чаще всего работа системы, полученной по циклограмме в соответствии с формулой (15), не удовлетворяет технологическим тре-

бованиям, заданию на функционирование. В выражения приходится вводить промежуточные элементы или элементы «самоблокировки», т.е. необходимо, чтобы в схеме была «память».

Для использования полученных выражений и в соответствии с методом циклограмм необходимо выполнить четыре проверки.

Первая проверка заключается в анализе того, существует ли записанное ранее условие включения $f'_{(x_i)}$в течение всего включающего периода. Если $f'_{(x_i)} = const$, то условие включения является достаточным. Если $f'_{(x_i)}$ изменяет свое состояние в течение включающего периода, то необходимо обеспечить самоблокировку:

$$X = f'_{(x_i)} + X \cdot \overline{f''_{(x_i)}} \tag{16}$$

Вторая проверка предназначена для выявления того, существует ли записанное ранее условие отключения $f''_{(x_i)}$в течение всего периода включения. Если $f''_{(x_i)} = const$, то условие отключения для данного периода является достаточным. Если $f''_{(x_i)}$изменяет свое состояние, то необходимо ввести промежуточный элемент $p''_{(x)}$:

$$X = f'_{(x_i)} \cdot \overline{p''_{(x)} \cdot f''_{(x_i)}} \tag{17}$$

Третья проверка предназначена для контроля того, чтобы после отключения выходного элемента не были бы созданы вновь условия для его непроизвольного повторного включения. С этой целью надо проверить, не присутствуют ли в выражениях для выходных сигналов комбинации следующих видов:

$$f'_{(x_i)} \cdot \overline{f''_{(x_i)}},\ f'_{(x_i)} + X \cdot \overline{f''_{(x_i)}},\ f'_{(x_i)} \cdot \overline{p''_{(x)} \cdot f''_{(x_i)}},\ f'_{(x_i)} + X \cdot \overline{p''_{(x)} \cdot f''_{(x_i)}}.$$

Если встречается одна из этих комбинаций, то вводится промежуточный элемент $p'''_{(x)}$. В качестве $p''_{(x)}$ и $p'''_{(x)}$желательно использовать другие входные, выходные или уже введенные ранее промежуточные элементы, изменяющие свое состояние так, как необходимо с точки зрения срабатывания либо несрабатывания рассматриваемого выходного элемента.

После того, как по циклограмме для каждого цикла с учетом трех проверок реализуемости циклограмм составляются функциональные выражения, выполняется четвертая проверка, которая должна установить, не встречаются ли одинаковые комбинации пе-

ременных в циклограммах разных циклов. Если они встречаются, то вводятся дополнительные элементы, как и при третьей проверке.

Когда проведены все четыре проверки, записываются формулы для каждого из выходных элементов для всех периодов включения (циклов):

$$X = X_1 + X_1 + \cdots + X_n \qquad (18)$$

3.2. Проектирование и расчет управляющих логических устройств

3.2.1. Построение циклограммы

В качестве примера приведем расчет и спроектируем управляющее логическое устройство на базе ПЛИС (*PLD*, ***P****rogrammable* ***L****ogic* ***D****evice*) в которых в отличие от обычных цифровых микросхем логика работы не определяется при изготовлении, а задается посредством программирования (проектирования). Для программирования используется программатор и интегрированная среда разработки (*IDE*, ***I****ntegrated* ***D****evelopment* ***E****nvironment*), позволяющие задать желаемую структуру цифрового устройства в виде принципиальной электрической схемы или программы на специальных языках описания аппаратуры *Verilog*, *VHDL* и др. Главным отличием ПЛИС от микроконтроллеров является то, что в микроконтроллере нельзя изменять внутренние связи между простейшими элементами, а в ПЛИС прописывания связей является основой для программирования и работы с ними. Кроме этого, микроконтроллер выполняет последовательно все операции, прописанные в его программе, в то время как блоки ПЛИС выполняют задачу параллельно и независимо друг от друга.

Поскольку любая логическая функция может быть представлена в виде суммы произведений – дизъюнктивной нормальной формы, базовыми структурными компонентами ПЛИС являются матрицы элементов «И» и «ИЛИ». ПЛИС также содержат многочисленные обратные связи (ОС), позволяющие использовать текущие состояния и формировать последовательностные автоматы различных классов.

Ниже на рис.5 приведена таблица включений УЛУ, а на рис.6 – полученная по таблице включений циклограмма.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42
a	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
b	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	1
c	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	1	0	0
d	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	1
e	1	1	1	0	0	1	1	0	1	1	0	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	0	0	1	1
f	0	0	1	0	0	0	1	0	0	1	0	1	1	1	0	1	1	1	0	0	1	1	0	0	1	1	0	1	1	0	0	1	1	0	0	1	0	1	1	1	1	0
g	1	1	1	0	1	1	1	1	0	1	1	0	1	0	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	0	0	1	1	1	1	0	0	1	0	1	1	0
h	0	1	0	0	0	1	0	1	0	0	1	0	1	0	0	1	0	0	0	1	0	0	1	0	0	0	1	0	0	1	0	1	0	1	0	1	1	1	0	0	1	1
i	1	1	1	1	0	1	1	0	1	1	1	0	0	1	1	1	1	0	1	1	1	1	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	0	1
X	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
Y	0	1	1	1	1	1	1	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	0
Z	0	0	1	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1

Рис.5. Таблица включений УЛУ

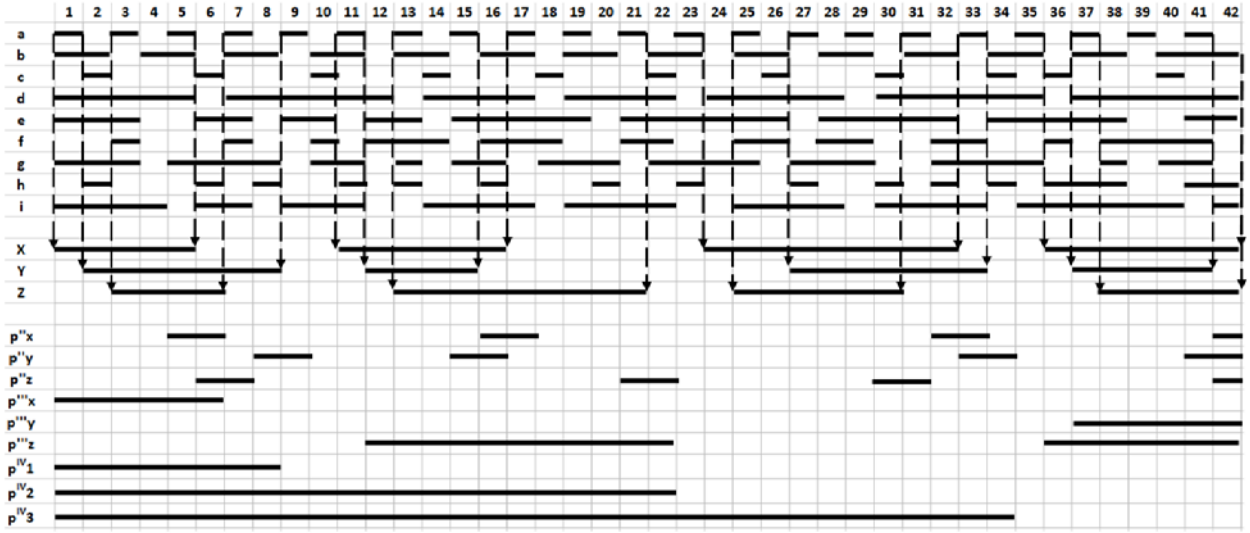

Рис.6. Циклограмма работы УЛУ

При построении циклограммы предполагается, что включение и отключение выходных элементов осуществляется теми входными элементами, которые изменяют свое значение в тех же тактах, что и выходные элементы.

3.2.2. Определение функциональных выражений

По полученной циклограмме с учетом проверок определяем функциональные выражения для выходных сигналов *X*, *Y*, *Z*.

Функциональные выражения для сигнала *X*:

$$f_{x1}^{'} = a \cdot b \cdot d \cdot e \cdot g \cdot i;\ f_{x1}^{''} = \bar{a} + \bar{b} + c + \bar{d} + e + h + i;$$
$$x_1 = (a \cdot b \cdot d \cdot e \cdot g \cdot i + x_1 \cdot \overline{p_x^{''} \cdot (\bar{a} + \bar{b} + c + \bar{d} + e + h + \imath)})p_x^{'''};$$

$$f_{x2}^{'} = a \cdot \bar{c} \cdot \bar{e} \cdot \bar{f} \cdot h;\ f_{x2}^{''} = a + \bar{g} + \bar{h};$$
$$x_2 = a \cdot \bar{c} \cdot \bar{e} \cdot \bar{f} \cdot h + x_2 \cdot \overline{p_x^{''} \cdot (a + \bar{g} + \bar{h})};$$

$$f_{x3}^{'} = \bar{a} \cdot \bar{b} \cdot d \cdot \bar{h};\ f_{x3}^{''} = a + \bar{b} + \bar{e} + \bar{h};$$
$$x_3 = \bar{a} \cdot \bar{b} \cdot d \cdot \bar{h} + x_3 \cdot \overline{p_x^{''} \cdot (a + \bar{b} + \bar{e} + \bar{h})};$$

$$f_{x4}^{'} = \bar{a} \cdot \bar{b} \cdot c \cdot \bar{d} \cdot f \cdot \bar{g} \cdot h;\ f_{x4}^{''} = \bar{b} + \bar{d} + \bar{e} + \bar{h} + \bar{\imath};$$
$$x_4 = \bar{a} \cdot \bar{b} \cdot c \cdot \bar{d} \cdot f \cdot \bar{g} \cdot h + x_4 \cdot \overline{p_x^{''} \cdot (\bar{b} + \bar{d} + \bar{e} + \bar{h} + \bar{\imath})};$$

Окончательно для *X* получаем:

$$X = x_1 \cdot p_1^{IV} \cdot p_2^{IV} \cdot p_3^{IV} + x_2 \cdot \overline{p_1^{IV}} \cdot p_2^{IV} \cdot p_3^{IV} + x_3 \cdot \overline{p_1^{IV}} \cdot \overline{p_2^{IV}} \cdot p_3^{IV} +$$
$$+ x_4 \cdot \overline{p_1^{IV}} \cdot \overline{p_2^{IV}} \cdot \overline{p_3^{IV}}. \qquad (19)$$

Функциональные выражения для сигнала *Y*:

$$f_{y1}^{'} = \bar{a} \cdot c \cdot h;\ f_{y1}^{''} = a + \bar{b} + e + \bar{g} + \bar{h} + i;$$
$$y_1 = \bar{a} \cdot c \cdot h + y_1 \cdot \overline{p_y^{''} \cdot (a + \bar{b} + e + \bar{g} + \bar{h} + \imath)};$$

$$f_{y2}^{'} = \bar{a} \cdot \bar{b} \cdot e \cdot f \cdot \bar{g} \cdot \bar{h} \cdot \bar{\imath};\ f_{y2}^{''} = \bar{a} + b + f + h;$$

$$y_2 = \bar{a} \cdot \bar{b} \cdot e \cdot f \cdot \bar{g} \cdot \bar{h} \cdot \bar{\imath} + y_2 \cdot \overline{p_y^{''} \cdot (\bar{a} + b + f + h)};$$

$$f_{y3}^{'} = a \cdot \bar{b} \cdot \bar{c} \cdot \bar{e} \cdot \bar{f} \cdot g \cdot h;\ f_{y3}^{''} = \bar{a} + b + c + e + \bar{f} + h + \bar{\imath};$$
$$y_3 = a \cdot \bar{b} \cdot \bar{c} \cdot \bar{e} \cdot \bar{f} \cdot g \cdot h + y_3 \cdot \overline{p_y^{''} \cdot \left(\bar{a} + b + c + e + \bar{f} + h + \bar{\imath}\right)};$$

$$f_{y4}^{'} = a \cdot b \cdot \bar{c} \cdot d \cdot \bar{f};\ f_{y4}^{''} = \bar{a} + \bar{f} + \bar{g} + i;$$
$$y_4 = a \cdot b \cdot \bar{c} \cdot d \cdot \bar{f} + y_4 \cdot \overline{p_y^{''} \cdot \left(\bar{a} + \bar{f} + \bar{g} + \imath\right)} p_y^{'''};$$

Окончательно для *Y* получаем:

$$Y = y_1 \cdot p_1^{IV} \cdot p_2^{IV} \cdot p_3^{IV} + y_2 \cdot \overline{p_1^{IV}} \cdot p_2^{IV} \cdot p_3^{IV} + y_3 \cdot \overline{p_1^{IV}} \cdot \overline{p_2^{IV}} \cdot p_3^{IV} +$$
$$+ y_4 \cdot \overline{p_1^{IV}} \cdot \overline{p_2^{IV}} \cdot \overline{p_3^{IV}}. \qquad (20)$$

Функциональные выражения для сигнала *Z*:

$$f_{z1}^{'} = a \cdot \bar{b} \cdot \bar{c} \cdot f \cdot \bar{h};\ f_{z1}^{''} = a + b + \bar{c} + d + f + \bar{h};$$
$$z_1 = a \cdot \bar{b} \cdot \bar{c} \cdot f \cdot \bar{h} + z_1 \cdot \overline{p_z^{''} \cdot \left(a + b + \bar{c} + d + f + \bar{h}\right)};$$

$$f_{z2}^{'} = a \cdot b \cdot \bar{d} \cdot g \cdot h;\ f_{z2}^{''} = \bar{a} + b + c + g;$$
$$z_2 = (a \cdot b \cdot \bar{d} \cdot g \cdot h + z_2 \cdot \overline{p_z^{''} \cdot (\bar{a} + b + c + g)}) p_z^{'''};$$

$$f_{z3}^{'} = a \cdot b \cdot f \cdot i;\ f_{z3}^{''} = a + b + \bar{c} + \bar{h};$$
$$z_3 = a \cdot b \cdot f \cdot i + z_3 \cdot \overline{p_z^{''} \cdot \left(a + b + \bar{c} + \bar{h}\right)};$$

$$f_{z4}^{'} = \bar{a} \cdot f \cdot g;\ f_{z4}^{''} = \bar{b} + \bar{d} + \bar{e} + \bar{h} + \bar{\imath};$$
$$z_4 = (\bar{a} \cdot f \cdot g + z_4 \cdot \overline{p_z^{''} \cdot \left(\bar{b} + \bar{d} + \bar{e} + \bar{h} + \bar{\imath}\right)}) p_z^{'''};$$

Окончательно для *Z* получаем:

$$Z = z_1 \cdot p_1^{IV} \cdot p_2^{IV} \cdot p_3^{IV} + z_2 \cdot \overline{p_1^{IV}} \cdot p_2^{IV} \cdot p_3^{IV} + z_3 \cdot \overline{p_1^{IV}} \cdot \overline{p_2^{IV}} \cdot p_3^{IV} +$$
$$+ z_4 \cdot \overline{p_1^{IV}} \cdot \overline{p_2^{IV}} \cdot \overline{p_3^{IV}}.$$
(21)

3.2.3. Реализация УЛУ на базе ПЛИС

На основе полученных функциональных выражений выходных параметров выполняется реализация УЛУ на базе ПЛИС в программной среде *MAX+PLUS II.*

САПР *MAX+ PLUS II* представляет собой интегрированную среду для разработки цифровых устройств на базе ПЛИС фирмы *Altera* и обеспечивает выполнение всех этапов, необходимых для выпуска готовых изделий:

- создание проектов устройств;
- синтез структур и трассировку внутренних связей ПЛИС;
- подготовку данных для программирования или конфигурирования ПЛИС (компиляцию);
- верификацию проектов (функциональное моделирование и временной анализ);
- программирование или конфигурирование ПЛИС.

В состав пакета *MAX+ PLUS II* входят несколько приложений, реализующих все перечисленные выше этапы разработки цифровых устройств на основе ПЛИС:

Graphic Editor – графический редактор, предназначенный для ввода проекта в виде схемы соединений символов элементов, извлекаемых из стандартных библиотек пакета либо из библиотеки пользователя;

Waveform Editor – редактор временных диаграмм (сигнальный редактор), который выполняет двойную функцию: на этапе ввода обеспечивает ввод логики проекта в виде диаграмм (эпюр) состояний входов и выходов, а на этапе моделирования обеспечивает ввод диаграмм тестовых (эталонных) входных состояний моделируемого устройства и задание перечня тестируемых выходов.

Text Editor – текстовый редактор, предназначенный для создания и редактирования текстовых файлов, содержащих описание логики проекта на языке описания устройств *AHDL* (*Altera Hardware Description Language*) или на близких к нему языках типа *VHDL*, *Verilog*.

Symbol Editor – символьный редактор, позволяющий редактировать существующие символы и создавать новые. Любой откомпилированный проект может быть свёрнут в символ, помещён в библиотеку символов и использован как элемент в любом другом проекте.

Floorplan Editor – редактор связей (поуровневый планировщик), который на плане расположения основных логических элементов позволяет вручную распределять выводы ПЛИС (закреплять выводы за конкретными входными и выходными сигналами) и перераспределять некоторые внутренние ресурсы ПЛИС.

Перед тем как начать работать в системе *MAX+PLUS II*, следует понять разницу между файлами проекта, вспомогательными файлами и проектами.

Файл проекта – это графический, текстовый или сигнальный файл, созданный с помощью графического или сигнального редакторов системы *MAX+PLUS II* или в любом другом, использующем промышленные стандарты, схемном или текстовом редакторе либо при помощи программы *netlist writer*, имеющейся в пакетах, поддерживающих *VHDL и Verilog HDL*. Этот файл содержит логику для проекта *MAX+PLUS II* и компилируется компилятором. Компилятор может автоматически обрабатывать следующие файлы проекта: графические файлы проекта (*.gdf*); текстовые файлы проекта на языке *AHDL* (*.tdf*); сигнальные файлы проекта (*.wdf*); файлы проекта на языке *VHDL* (*.vhd*); файлы проекта на языке *Verilog* (*.v*); схемные файлы *OrCAD* (*.sch*); входные файлы *EDIF* (*edf*); файлы формата *Xilinx Netlist* (*.xnf*); файлы проекта *Altera* (*.adf*).

Вспомогательные файлы – это файлы, связанные с проектом *MAX+PLUS II*, но не являющиеся частью его иерархического дерева. Большинство таких файлов не содержит логики проекта. Некоторые из них создаются автоматически приложением системы *MAX+PLUS* II, другие - пользователем. Примерами вспомогательных файлов являются файлы назначений и конфигурации (*.acf*), символьные файлы (*.sym*), файлы отчета (*.rpt*) и файлы тестовых векторов (*.vec*).

Проект состоит из всех файлов иерархической структуры проекта, в том числе вспомогательных и выходных файлов. Именем проекта является имя файла проекта верхнего уровня без расшире-

ния. Система *MAX+PLUS II* выполняет компиляцию, тестирование, анализ синхронизации и программирование сразу целого проекта, хотя пользователь может в это время редактировать файлы этого проекта в рамках другого проекта. Для каждого проекта желательно создавать отдельный подкаталог в рабочем каталоге *MAX+PLUS II.*

В системе *MAX+PLUS II* разработка проекта ускоряется за счёт имеющихся стандартных функций, в том числе примитивов, мегафункций и библиотеки параметризованных модулей *LPM.* В иерархической структуре проекта на любом уровне допускается смешанное использование файлов с расширениями *".gdf .tdf .vhd .v .edf .sch".* Однако файлы с расширением *".wdf .xnf .adf .smf "* должны быть либо на самом нижнем иерархическом уровне проекта, либо быть единственными.

После выполнения всех назначений и задания проекта приступают к его компиляции. Сначала компилятор извлекает информацию об иерархических связях между файлами проекта и проверяет проект на простые ошибки ввода описания проекта. Компилятор применяет разнообразные способы увеличения эффективности проекта и минимизации использования ресурсов устройства. Если проект слишком большой, чтобы быть реализованным в одном устройстве, компилятор может автоматически разбить его на части для реализации в нескольких устройствах того же самого семейства, при этом число соединений между устройствами минимизируется.

Кроме того, компилятор создает программирующие файлы, используемые программатором для программирования одного или нескольких устройств. У разработчика также есть возможность настроить обработку проекта. Например, можно задать стиль логического синтеза проекта по умолчанию и другие параметры логического синтеза в рамках всего проекта, что позволит провести логический синтез в соответствии с вашими потребностями. Кроме того, вы можете ввести требования по синхронизации в рамках всего проекта, точно задать разбиение большого проекта на части для реализации в нескольких устройствах и выбрать варианты параметров устройств, которые будут применены для всего проекта в целом.

На рис.7–9, приведены схемы для выходных сигналов, полученные по функциональным выражениям (19–21), а на рис. 10–12 – временные диаграммы для этих сигналов.

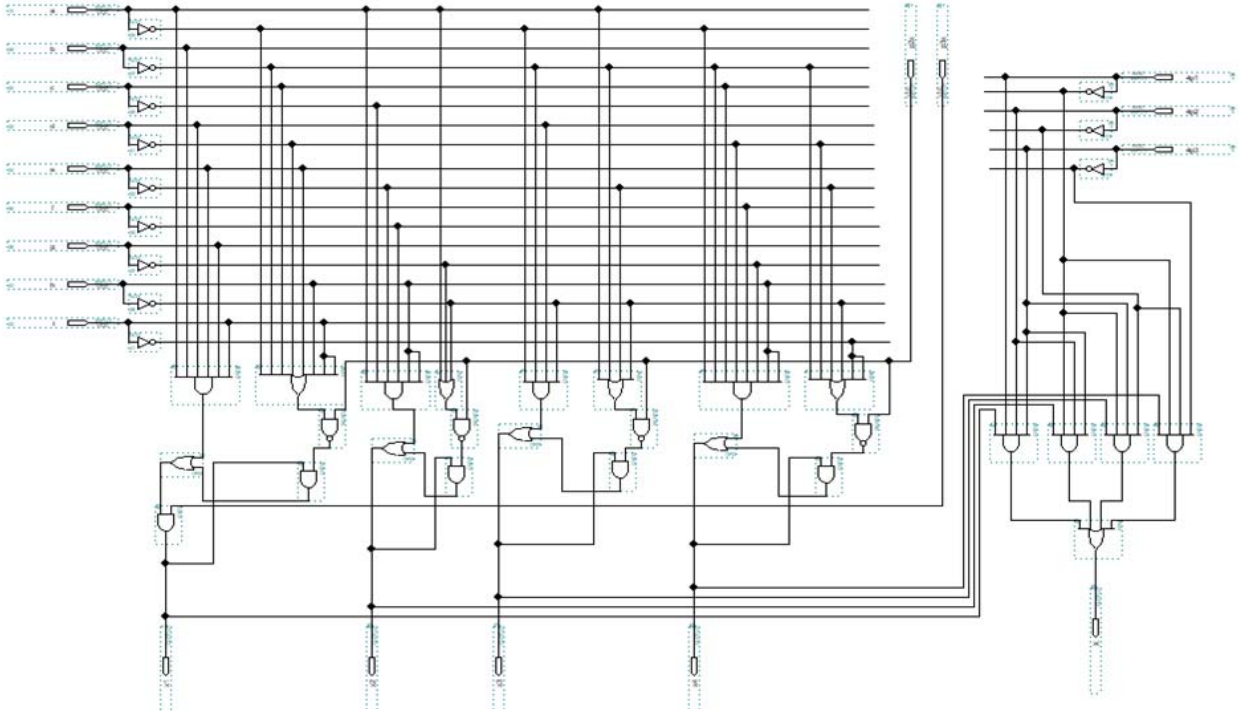

Рис.7. Логическая схема УЛУ для выходного сигнала *X*.

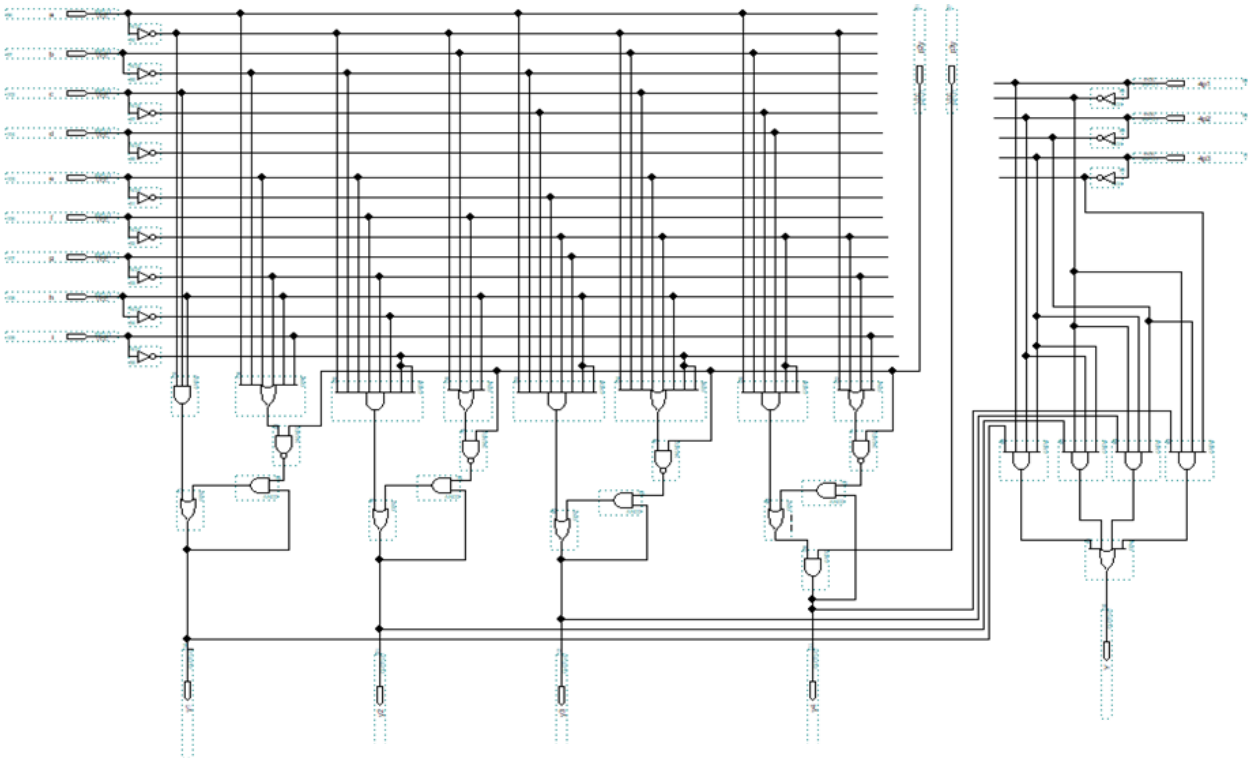

Рис.8. Логическая схема УЛУ для выходного сигнала *Y*.

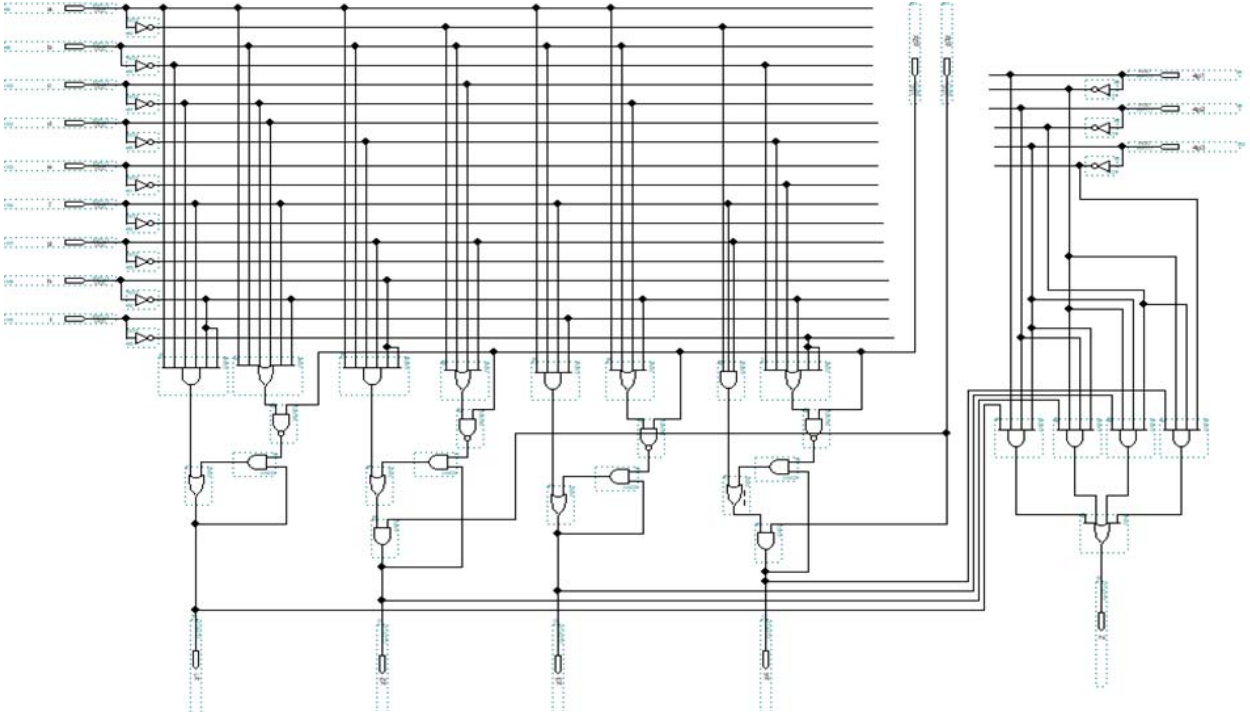

Рис.9. Логическая схема УЛУ для выходного сигнала *Z*.

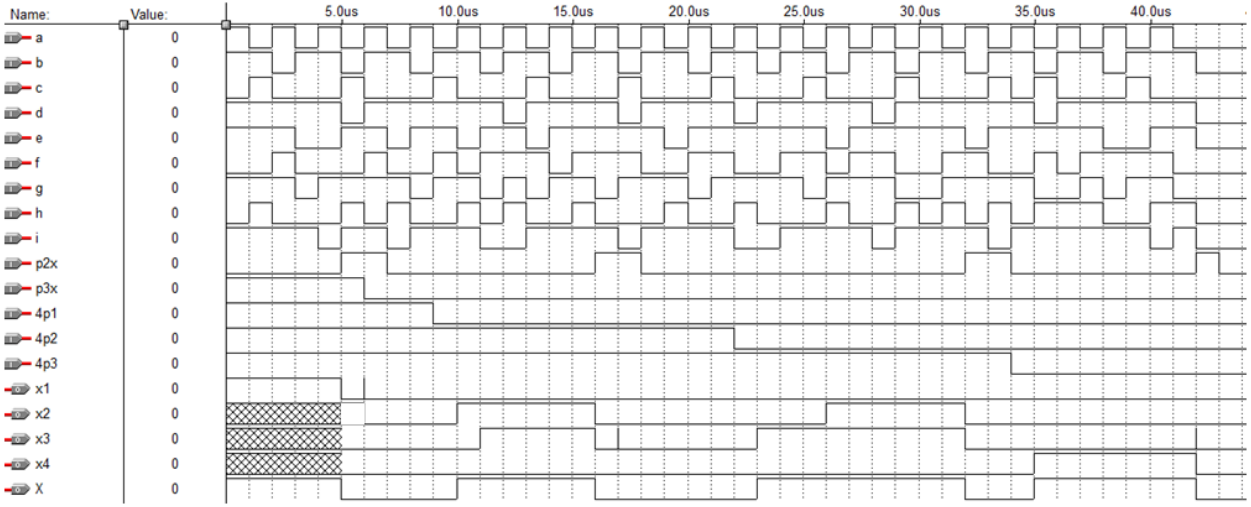

Рис.10. Временная диаграмма для выходного сигнала *X*.

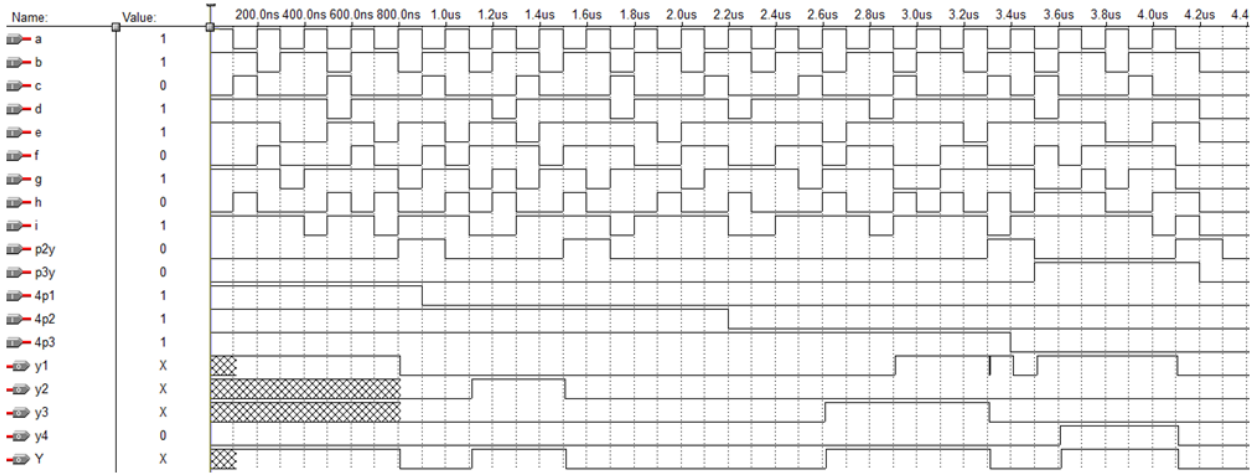

Рис.11. Временная диаграмма для выходного сигнала *Y*.

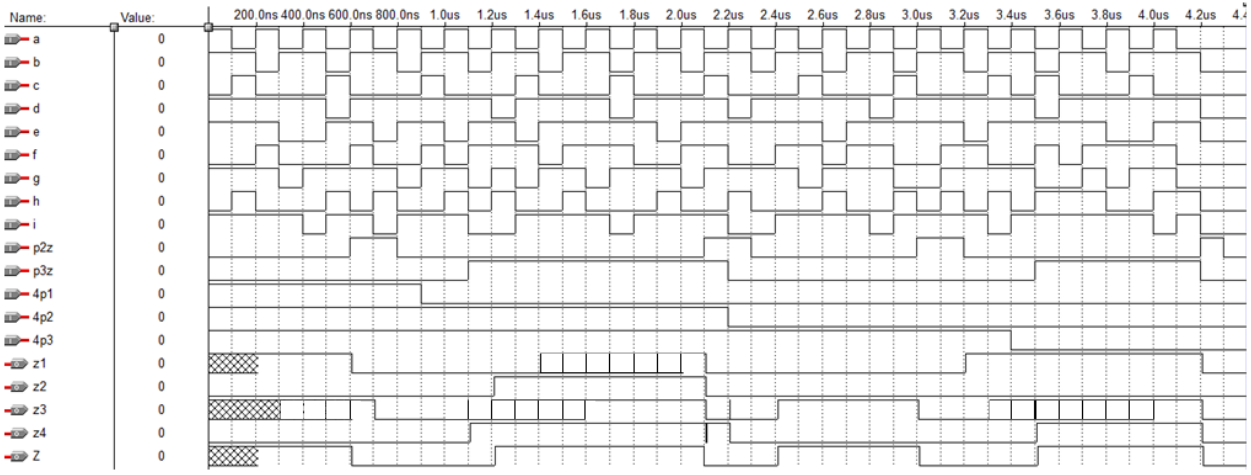

Рис.12. Временная диаграмма для выходного сигнала *Z*.

По результатам компиляции схем в *MAX+PLUS II* процессор сообщений информирует о том, что для реализации данного проекта может быть использована ПЛИС типа EP1K10TC100–1 (рис.13).

Рис.13. Результат компиляции в *MAX+PLUS II*.

Полученные циклограммы сигналов в программе *MAX+PLUS II* полностью совпали с исходной циклограммой, поэтому можно приступить к следующему этапу – разработке печатной платы.

3.2.4. Проектирование печатной платы

Для разработки схемных компонентов, посадочных мест и создания фотошаблона печатной платы используются инструментарии пакета САПР *DipTrace*. В состав программного пакета *DipTrace* входят следующие модули (рис.14):

- *Schematic Capture*. Это среда для разработки принципиальной схемы. Они могут быть простыми и многоуровневыми со сложными связями между блоками. Здесь присутствуют опции проверки иерархий, связи и так далее.
- *PCB Layout*. Инструмент для разработки печатных плат в автоматическом или ручном режиме. Есть возможность размещать компоненты и подключать внешние трассировщики.
- *Component Editor*. Инструмент для создания и обработки библиотеки компонентов. Есть возможность создавать многосекционные компоненты. При этом использовать шаблоны и другие параметры.
- *Pattern Editor*. Редактор корпусов компонентов. Здесь можно создать корпуса разных форм, контактных площадок. Имеется большой выбор готовых шаблонов.

Рис.14. Программные модули *DipTrace*.

Стандартная библиотека содержит около 100 тысяч компонентов. С помощью 3*D* просмотра есть возможность просматривать трехмерную модель платы. Система позволяет импортировать схемы из различных приложений. Основные возможности программы: ручное и автоматическое позиционирование, эффективная трассировка, всесторонняя проверка проекта, моделирование существующей схемы, создание файла для производства, создание личных библиотек.

DipTrace содержит минимально возможное количество управляющих элементов, при работе редактируемые объекты подсвечиваются, что позволяет наглядно оценивать ситуацию. Изменение одного элемента схемы или платы отражается на всех зависящих от него объектах. Автотрассировщик может работать со сложными многослойными платами, имеющими различные типы радиодеталей. Программа проводит многочисленные проверки проекта (новых элементов в библиотеке, допустимости и целостности соединений, зазоров, размерностей) на разных этапах работы, что позволяет обнаружить и исправить ошибки «на лету».

Рис.15. Микросхема EP1K10TC100-1

На рис.15 приведено изображение микросхемы ПЛИС EP1K10TC100-1 фирмы *Altera*. В *Schematic Capture* находим рекомендуемую схему ПЛИС и формируем соответствующие связи (рис.16). Далее разрабатывается печатная плата.

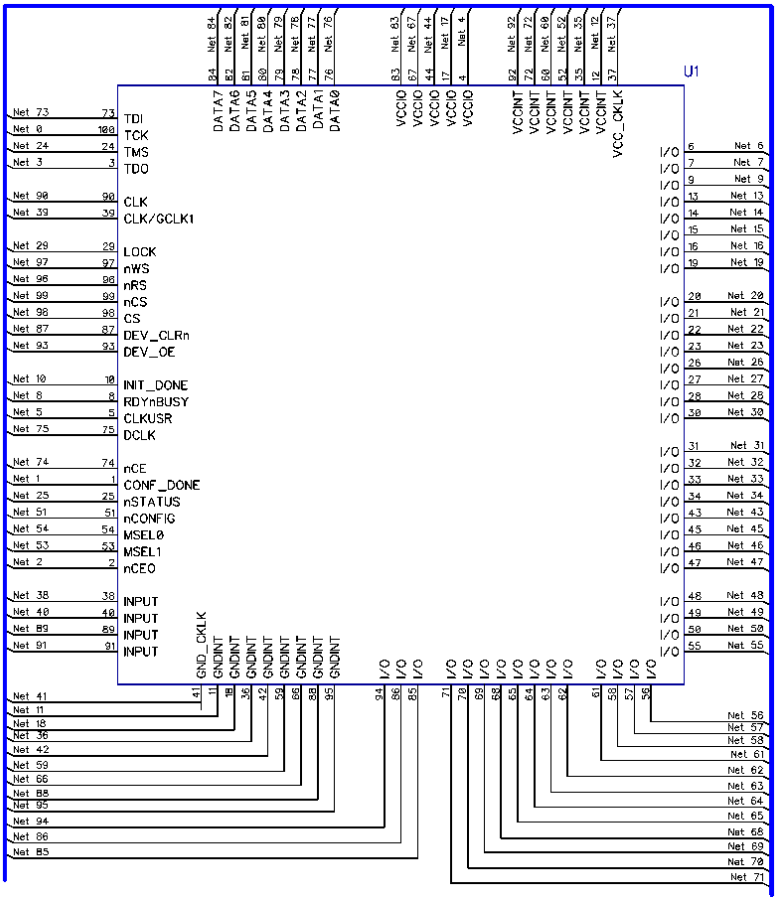

Рис.16. Схема ПЛИС EP1K10TC100-1

3.2.5. Программная реализация УЛУ

На рис.17 в качестве примера показана блок-схема алгоритма функционирования управляющего логического устройства, описываемого выражениями (19–21), а ниже – текст программы на VBA Excel, позволяющей реализовать УЛУ программным способом.

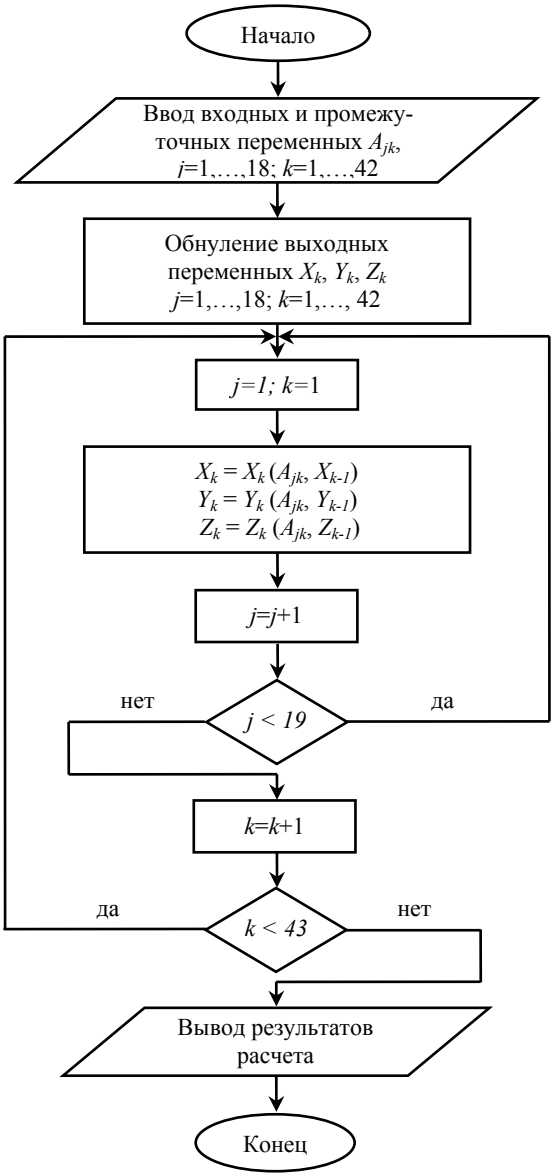

Рис.17. Блок-схема алгоритма работы УЛУ

Листинг программы

```
Private Sub Вычислить_Click()
Dim a As Variant, i, j, x1, x2, x3, x4, X, y1, y2, y3, y4, Y, z1, z2, z3, z4, Z As
Long
a = Range("B2:AQ20")
For j = 1 To 19
For i = 1 To 42
If a(j, i) = 1 Then Cells(j + 1, i + 1).Interior.Color = vbGreen
x1 = (a(1, i) And a(2, i) And a(4, i) And a(5, i) And a(7, i) And a(9, i) Or x1 And
Not (a(11, i) And (Not a(1, i) Or Not a(2, i) Or a(3, i) Or Not a(4, i) Or a(5, i) Or
a(8, i) Or a(9, i)))) And a(14, i)

x2 = a(1, i) And Not a(3, i) And Not a(5, i) And Not a(6, i) And a(8, i) Or
x2 And Not (a(11, i) And (a(1, i) Or Not a(7, i) Or Not a(8, i)))

x3 = Not a(1, i) And Not a(2, i) And a(4, i) And Not a(8, i) Or x3 And
Not (a(11, i) And (a(1, i) Or Not a(2, i) Or Not a(5, i) Or Not a(8, i)))

x4 = Not a(1, i) And Not a(2, i) And a(3, i) And Not a(4, i) And a(6, i) And Not
a(7, i) And a(8, i) Or x4 And Not (a(11, i) And (Not a(2, i) Or Not a(4, i) Or Not
a(5, i) Or Not a(8, i) Or Not a(9, i)))

X = x1 And a(17, i) And a(18, i) And a(19, i) Or x2 And Not a(17, i)
And a(18, i) And a(19, i) Or x3 And Not a(17, i) And Not a(18, i) And a(19, i)
Or x4 And Not a(17, i) And Not a(18, i) And Not a(19, i)
Cells(22, i + 1) = X
If X = 1 Then Cells(22, i + 1).Interior.Color = vbGreen
y1 = Not a(1, i) And a(3, i) And a(8, i) Or y1 And Not (a(12, i) And (a(1, i)
Or Not a(2, i) Or a(5, i) Or Not a(7, i) Or a(8, i) Or a(9, i)))

y2 = Not a(1, i) And Not a(2, i) And a(5, i) And a(6, i) And Not a(7, i) And Not
a(8, i) And Not a(9, i) Or y2 And Not (a(12, i) And (Not a(1, i) Or a(2, i)
Or a(6, i) Or a(8, i)))

y3 = a(1, i) And Not a(2, i) And Not a(3, i) And Not a(5, i) And Not a(6, i) And
a(7, i) And a(8, i) Or y3 And Not (a(12, i) And (Not a(1, i) Or a(2, i) Or
a(3, i) Or a(5, i) Or Not a(6, i) Or a(8, i) Or Not a(9, i)))
y4 = (a(1, i) And a(2, i) And Not a(3, i) And a(4, i) And Not a(6, i) Or y4 And
Not (a(12, i) And (Not a(1, i) Or Not a(6, i) Or Not a(7, i) Or a(9, i))))
```

```
And a(15, i)
Y = y1 And a(17, i) And a(18, i) And a(19, i) Or y2 And Not a(17, i) And a(18,
i) And a(19, i) Or y3 And Not a(17, i) And Not a(18, i) And a(19, i) Or y4 And
Not a(17, i) And Not a(18, i) And Not a(19, i)
Cells(23, i + 1) = Y
   If Y = 1 Then Cells(23, i + 1).Interior.Color = vbGreen
z1 = a(1, i) And Not a(2, i) And Not a(3, i) And a(6, i) And Not a(8, i) Or z1 And
Not (a(13, i) And (a(1, i) Or a(2, i) Or Not a(3, i) Or a(4, i) Or a(6, i) Or Not a(8,
i)))

z2 = (a(1, i) And a(2, i) And Not a(4, i) And a(7, i) And a(8, i) Or z2 And Not
(a(13, i) And (Not a(1, i) Or a(2, i) Or a(3, i) Or a(7, i)))) And a(16, i)

z3 = a(1, i) And a(2, i) And a(6, i) And a(9, i) Or z3 And Not (a(13, i)
And (a(1, i) Or a(2, i) Or Not a(3, i) Or Not a(8, i)))

z4 = (Not a(1, i) And a(6, i) And a(7, i) Or z4 And Not (a(13, i) And (Not a(2, i)
Or Not a(4, i) Or Not a(5, i) Or Not a(8, i) Or Not a(9, i)))) And a(16, i)

Z = z1 And a(17, i) And a(18, i) And a(19, i) Or z2 And Not a(17, i) And a(18, i)
And a(19, i) Or z3 And Not a(17, i) And Not a(18, i) And a(19, i) Or z4 And Not
a(17, i) And Not a(18, i) And Not a(19, i)
Cells(24, i + 1) = Z
   If Z = 1 Then Cells(24, i + 1).Interior.Color = vbGreen
  Next i
  Next j
End Sub
Private Sub Заполнить_Click()
Dim a As Variant
a = Sheets("Лист2").Range("B2:AQ20")
Sheets("Лист1").Range("B2:AQ20") = a
End Sub
Private Sub Сбросить_Click()
Range("B2:AQ24").Interior.Color = vbWhite
Range("B2:AQ24").ClearContents
End Sub
```

Результаты работы программы представлены на рис.18.

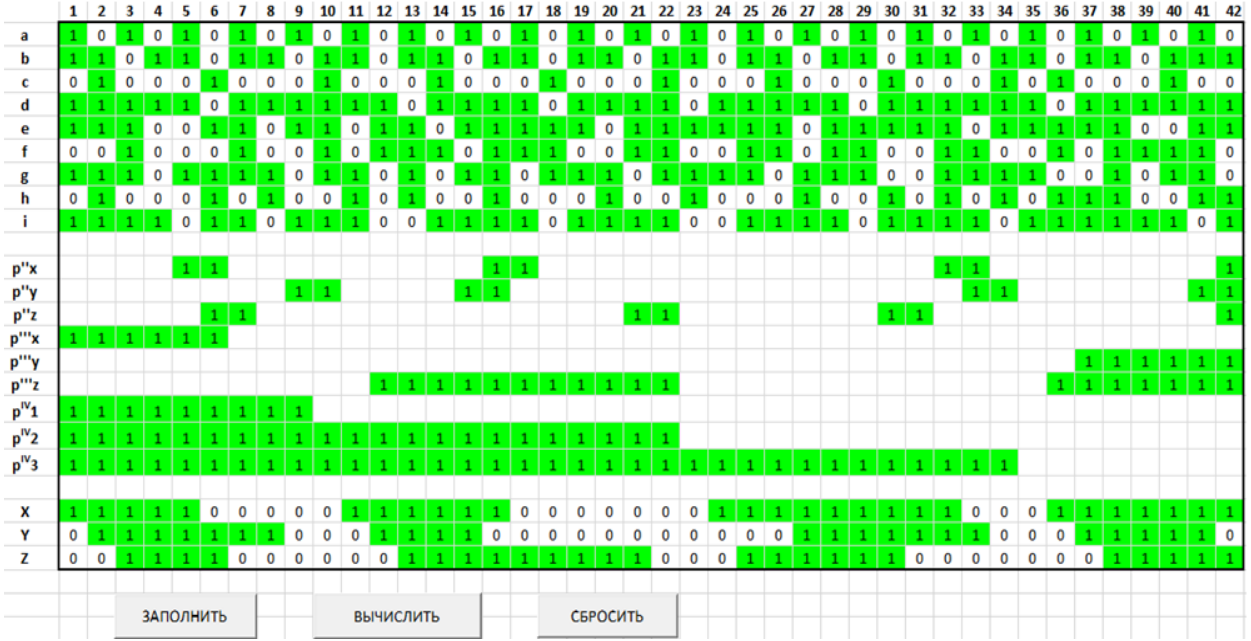

Рис18. Результат выполнения программы.

Видно, что таблицы включений, полученные при аппаратной и программной реализациях УЛУ для формул (19-21), идентичны, и соответствуют заданной, что в свою очередь подтверждает правильность выполненных построений при проектировании устройства.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

а) основная литература

1. *Кондаков А.И.*. САПР технологических процессов. М., Изд. «Академия». 2010. - 272 с.
2. *Стешенко В.Б.* ПЛИС фирмы «Altera»: элементная база, система проектирование и языки описания аппаратуры. М.: Изд. «ДМК-Пресс», 2015. - 576 с.
3. *Тарасов И.Е.* ПЛИС Xilinx. Языки описания аппаратуры VHDL и Verilog, САПР, приемы проектирования. М., Изд. «Горячая Линия - Телеком», 2019. - 538 с.
4. *Бельков В.Н., Ланшаков В.Л.* Автоматизированное проектирование технических систем: Учебное пособие. Изд. "Академия Естествознания", 2009. - 143 с.
5. *Ушаков Д.М.*. Введение в математические основы САПР. Изд. «ДМК-Пресс». 2011. - 208 с.
6. DipTrace. Руководство пользователя. 2016. https://diptrace.com/books/tutorial_rus.pdf.
7. *Джон Уокенбах. Excel* 2013. Профессиональное программирование на VBA. М., Изд. «Вильямс», 2017. - 960 с.

б) дополнительная литература

1 *Маларев В.И.* Системы автоматизированного проектирования. Учебное пособие. СПб, Изд. СПГГИ, 2000. - 52 с.
2. *Маларев В.И.* Проектирование и расчет систем автоматики. Учебное пособие. СПб., Изд. СПГГИ, 2003.- 88 с.
3. *Грейнер. Г.Р.* Проектирование бесконтактных управляющих логических устройств промышленной автоматики, М. Изд. «Энергия», 1977. - 384 с.
4. *Кунву Ли.* Основы САПР CAD/CAM/CAE. М. Изд. «Питер», 2004. – 560 с.

5. *Гольдбег О.Д., Свириденко И.С.* Инженерное проектирование и САПР электрических машин. Изд. «Академия». 2008. - 560 с.
6. *Жарков Н.В., Финков М.В.* AutoCAD 2019. Полное руководство. СПб, Изд. «Наука и техника СПб». 2019, – 640 с.
7. *Кечиев Л.Н.* Проектирование печатных плат для цифровой быстродействующей аппаратуры М., Изд. «Издательский Дом Технологии», 2007. - 617 с.
8. Системы автоматизированного проектирования электронных устройств и систем (E-CAD / EDA - системы): Учебное пособие / Под ред. Ю.В. Петрова; СПб, Изд. БГТУ, 2015. – 120 с.
9. *Попов А.Ю.* Проектирование цифровых устройств с использование ПЛИС: Учебное. пособие. М.: Изд. МГТУ им.Н.Э. Баумана, 2009. - 80 с.

СОДЕРЖАНИЕ