ИНФОРМАТИКА И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ / INFORMATICS AND INFORMATION PROCESSES

DOI: https://doi.org/10.60797/IRJ.2024.149.110

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ МОДЕЛИРОВАНИЯ РАБОТЫ ЦИФРОВЫХ СХЕМ С ИСПОЛЬЗОВАНИЕМ ПЛАТФОРМЫ .NET FRAMEWORK

Научная статья

Махфуд Б.А.[1, *], Пикулев А.Н.[2], Толмачева А.В.[3], Пирогова Т.П.[4]

[1, 2] Казанский национальный исследовательский технический университет им. А.Н. Туполева – КАИ, Казань, Российская Федерация

[3, 4] Казанский национальный исследовательский технологический университет, Казань, Российская Федерация

* Корреспондирующий автор (bmakhfud[at]kai.ru)

Аннотация

В данной работе представлено приложение для моделирования работы цифровых схем, разработанное на платформе .NET Framework с использованием языка программирования C#. Приложение ориентировано на использование в образовательных целях и призвано упростить процесс синтеза цифровых схем, предлагая удобный графический интерфейс и богатый набор стандартных компонентов. Основной задачей разработки было устранение недостатков существующих решений, таких как сложность использования и избыточность функционала. Приложение включает в себя инструменты для создания, редактирования и тестирования цифровых схем, что делает его удобным для студентов и преподавателей. Проведенные испытания подтвердили корректность работы всех элементов и функциональных возможностей программы.

Ключевые слова: схемотехника, моделирование цифровых схем, синтез цифровых схем, платформа .NET Framework, язык программирования C#.

DEVELOPMENT OF AN APPLICATION FOR MODELLING THE OPERATION OF DIGITAL CIRCUITS USING THE .NET FRAMEWORK PLATFORM

Research article

Makhfud B.A.[1, *], Pikulev A.N.[2], Tolmacheva A.V.[3], Pirogova T.P.[4]

[1, 2] Kazan National Research Technical University named after A.N. Tupolev – KAI, Kazan, Russian Federation

[3, 4] Kazan National Research Technological University, Kazan, Russian Federation

* Corresponding author (bmakhfud[at]kai.ru)

Abstract

This work presents an application for modelling the operation of digital circuits, developed on the .NET Framework platform using the C# programming language. The application is oriented for educational purposes and is designed to simplify the process of synthesizing digital circuits by offering a user-friendly graphical interface and a rich set of standard components. The main objective of the development was to eliminate the shortcomings of existing solutions, such as complexity of use and redundancy of functionality. The application includes tools for creating, editing and testing digital circuits, which makes it convenient for students and teachers. The conducted tests confirmed the correct operation of all elements and functionalities of the programme.

Keywords: circuit design, digital circuit modelling, digital circuit synthesis, .NET Framework, C# programming language.

Введение

Актуальность данной темы связана с тем, что в настоящее время обучающиеся Казанского национального исследовательского технического университета им. А.Н. Туполева – КАИ (КНИТУ-КАИ) по направлению 09.03.01 «Информатика и вычислительная техника» на начальных этапах обучения используют для моделирования последовательностных и комбинационных схем программу «ЭВЕМА», разработанную сотрудниками кафедры компьютерных систем КНИТУ-КАИ. Достоинствами данной программы являются простота в использовании, отсутствие лишнего функционала и низкий порог входа для моделирования схем. Однако она имеет и недостатки: для размещения элементов в рабочем поле и соединения проводников требуется излишнее количество действий, отсутствуют встроенные схемы наподобие дешифраторов, мультиплексоров и т.п.

Кроме «ЭВЕМА» существуют и другие программы, используемые для моделирования цифровых схем, такие как Multisim [1], [2] и Micro-Cap [3], [4]. Программа Multisim используется для моделирования электронных схем и предоставляет широкий набор возможностей для синтеза и анализа работы схем. Она поддерживает библиотеки различных элементов и моделирование различных измерительных приборов. Интерфейс программы Multisim интуитивно понятен и удобен для пользователей.

Micro-Cap – это программа для аналогового и цифрового моделирования электрических и электронных цепей с интегрированным визуальным редактором. Она позволяет моделировать практически любые логические схемы, но для работы с ней требуются знания в области электроники и схемотехники. Функционал программы оказывается избыточным для синтеза простых цифровых схем на уровне логических элементов и триггеров.

Каждое из рассмотренных решений имеет свои преимущества и недостатки. Программа «ЭВЕМА» проста в использовании, но требует много времени на выполнение базовых операций и не поддерживает стандартные

компоненты. Multisim и Micro-Cap предоставляют широкие возможности для моделирования сложных схем, но требуют глубоких знаний в электронике и могут быть сложными для начинающих пользователей.

Таким образом, существующие решения не полностью удовлетворяют потребности в эффективном синтезе простых цифровых схем [5], [6]. Необходимость разработки нового приложения для моделирования цифровых схем очевидна [7]. Оно должно объединять простоту использования и функциональность, чтобы ускорить процесс синтеза схем и сделать его более удобным для пользователей без глубоких знаний в электронике.

Практическая значимость данной работы заключается в том, что разработка приложения для моделирования работы цифровых схем, которое не будет иметь недостатков рассмотренных аналогов, позволит уменьшить время синтеза схемы за счет уменьшения количества действий, требуемых для размещения элементов, и расширения списка доступных компонентов, а также упростит их отладку благодаря возможности одновременного изменения состояний нескольких входных сигналов, триггеров.

Целью данной работы является разработка приложения для моделирования работы цифровых схем для использования в образовательном процессе.

Для достижения указанной цели поставлены задачи: разработка проекта приложения, его реализация и тестирование, а также обсуждение дальнейших перспектив развития разработанного программного модуля.

Проект приложения

Графический интерфейс программы не должен содержать большое количество элементов и должен быть понятным для пользователя, впервые запустившего её. В программе должны быть реализованы возможности сохранения схемы в файл и последующего восстановления для продолжения работы с того же места. Программа должна позволять пользователю добавлять элементы в любое место рабочей области, перемещать и удалять их.

В программе должны быть реализованы следующие элементы: логические вентили (И, ИЛИ, НЕ, И-НЕ, ИЛИ-НЕ), цифровой сигнал, генератор тактовых импульсов, триггеры (JK, D, T, RS), одноразрядный сумматор, дешифратор, мультиплексор, шифратор и демультиплексор [8], [9]. В программе должна быть реализована возможность изменять параметры уже существующих элементов. Программа должна поддерживать возможность изменения значения как одного логического сигнала или триггера, так и нескольких сразу.

Эти требования являются основополагающими для проектирования и разработки программного обеспечения, которое должно быть удобным и функциональным для моделирования цифровых схем. Важно обеспечить простоту использования, гибкость в работе с элементами схем, а также достаточный набор компонентов для создания различных логических конструкций.

Программа должна быть интуитивно понятной и поддерживать основные операции с элементами схемы, такие как добавление, перемещение, изменение параметров и удаление. Графический интерфейс должен быть минималистичным, чтобы не перегружать пользователя лишней информацией и элементами управления.

Графический интерфейс программы может выглядеть следующим образом: добавление элементов осуществляется нажатием клавиш определенных символов на клавиатуре: S — цифровой сигнал, G — генератор импульсов, A — логический вентиль И, O — логический вентиль ИЛИ, I — логический вентиль НЕ, D — дешифратор, C — шифратор, M — мультиплексор. Инвертирование логических вентилей или добавление демультиплексора осуществляется выбором соответствующего вентиля или мультиплексора и последующим нажатием клавиши SHIFT. Выбор количества входов для логических вентилей осуществляется нажатием цифровых клавиш 2-8.

На рисунке 1 представлен примерный макет графического интерфейса программы, демонстрирующий расположение основных элементов управления и области работы пользователя.

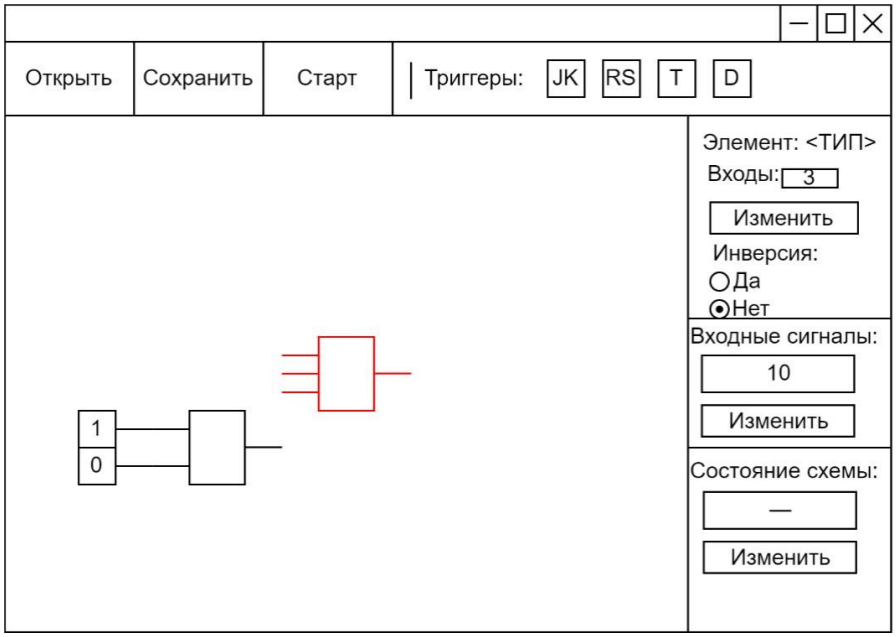

Рисунок 1 - Графический интерфейс программы

DOI: https://doi.org/10.60797/IRJ.2024.149.110.1

Выделим основные сущности приложения и определим их свойства. К основным сущностям относятся элемент, контакт и проводник. Под элементом будем понимать любой из компонентов, перечисленных в списке требований к разрабатываемому проекту.

Логический вентиль имеет набор входов (от одного до восьми) и один выход, который может быть инвертирован. Цифровой сигнал имеет один выход. Генератор тактовых импульсов также имеет один выход, период и коэффициент заполнения. Триггер имеет несколько входов, количество которых определяется типом триггера, и два выхода. Входы и выходы у триггера именованные. Одноразрядный сумматор имеет три входа и два выхода, которые также именованные. Дешифратор, мультиплексор, шифратор и демультиплексор имеют несколько входов и выходов, которые именованные. Свойства мультиплексора, шифратора и демультиплексора идентичны свойствам дешифратора.

Каждый элемент, за исключением цифрового сигнала и генератора, реализует некоторую функцию, которая ставит входным сигналам в соответствие не менее одного выходного сигнала. Таким образом, у всех элементов есть следующие свойства: список входных сигналов (возможно, пустой), список выходных сигналов, функция, определяющая значения выходных сигналов, списки имен входов и выходов (возможно, пустые). Наличие или отсутствие этих свойств у каждой из рассмотренных ранее сущностей указано в таблице.

Контакту соответствуют входы и выходы элементов. Контакт имеет следующие свойства: полярность (вход или выход) и текущее значение. Проводник обеспечивает передачу сигналов между контактами разной полярности. К свойствам проводника относятся контакт-источник (при наличии), контакт-приемник (при наличии) и список подключенных к данному проводнику других проводников.

Разрабатываемая программа предназначена для графического синтеза схем, поэтому к свойствам выделенных сущностей следует добавить атрибуты, характеризующие их положение в рабочем поле. Для элементов это будет прямоугольник с координатами (X; Y) верхней левой вершины, шириной W и высотой H. Для проводника таким атрибутом будет список точек с координатами (X_i; Y_i), где i — порядковый номер точки в списке. Для контакта в качестве таких атрибутов выступают пара точек P_1 и P_2 и координаты точки контакта: левой для контакта-входа и правой для контакта-выхода.

В процессе взаимодействия с программой пользователь постоянно совершает набор некоторых действий с целью получить необходимый ему результат. Набор таких действий будем называть сценарием, а необходимый результат – именем сценария. К основным сценариям взаимодействия отнесем размещение новых элементов, соединение существующих элементов, изменение параметров существующих элементов и удаление существующих элементов.

Рассмотрим данные сценарии, начиная со сценария «Размещение новых элементов» (см. рисунок 2). Чтобы добавить новый элемент на схему, пользователь нажимает соответствующую клавишу или кнопку в окне программы в случае, если требуется добавить триггер. Если выбранный элемент – логический вентиль, то пользователь также нажимает клавишу-цифру, соответствующую желаемому количеству входов. Если необходимо инвертировать логический вентиль, пользователь нажимает клавишу SHIFT. Далее пользователь выбирает положение элемента в рабочей области и нажимает левую кнопку мыши, после чего выбранный элемент будет размещен в рабочей области.

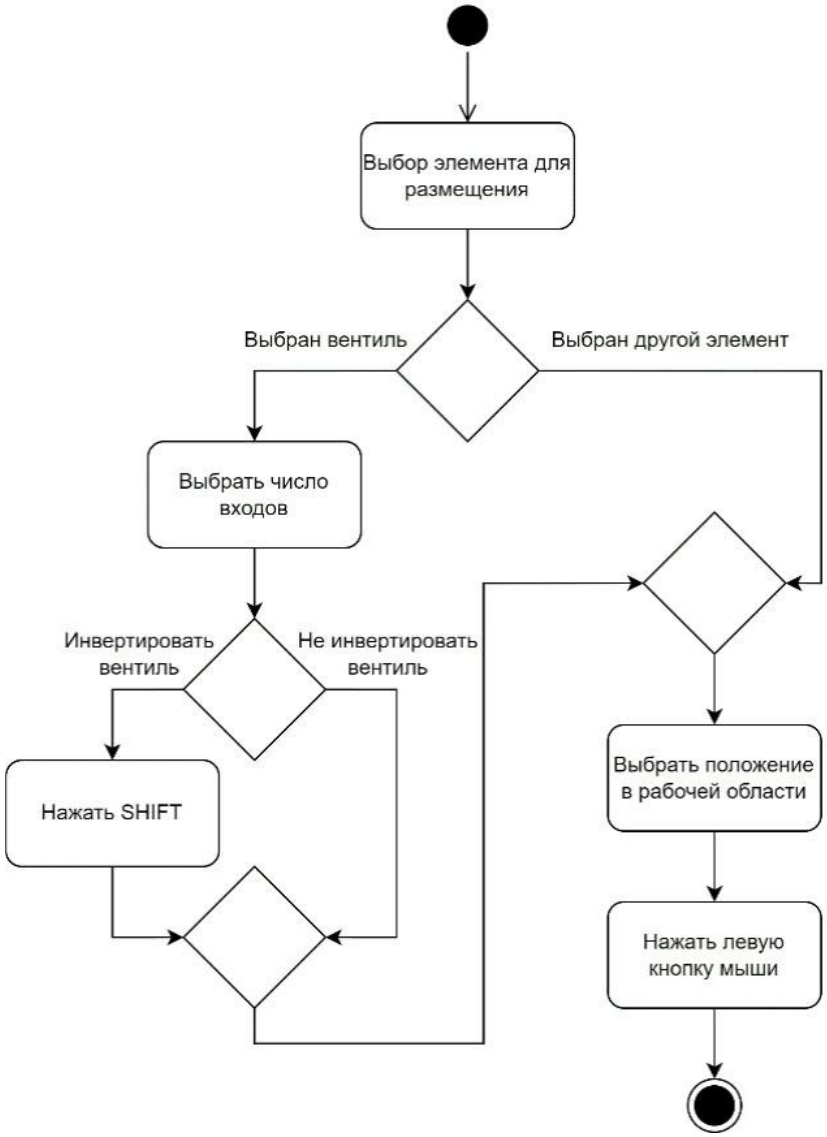

Рисунок 2 - Диаграмма деятельности для сценария «Добавить новый элемент»
DOI: https://doi.org/10.60797/IRJ.2024.149.110.2

Сценарий «Соединение элементов» подразумевает, что для соединения существующих элементов пользователь наводит указатель мыши на первый нужный контакт одного из соединяемых элементов, затем нажимает левую кнопку мыши (см. рисунок 3). Далее пользователь последовательно выбирает несколько точек, которые образуют линию проводника. На следующем шаге пользователь наводит указатель мыши на второй нужный контакт, противоположной относительно первого полярности, другого элемента.

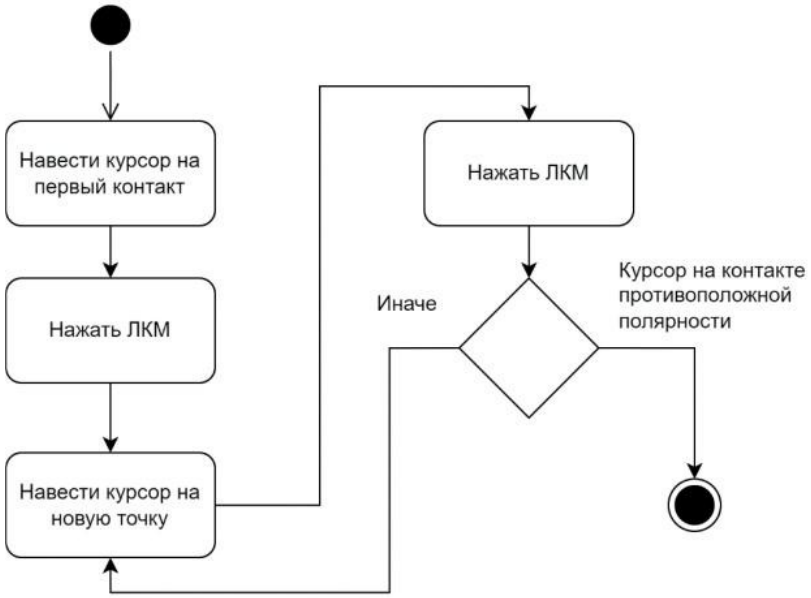

Рисунок 3 - Диаграмма деятельности для сценария «Соединение элементов»
DOI: https://doi.org/10.60797/IRJ.2024.149.110.3

Для изменения параметров элемента в сценарии «Изменение параметров элемента» пользователь наводит указатель мыши на нужный элемент, затем нажимает правую кнопку мыши (см. рисунок 4). Элемент становится выделенным. Затем пользователь устанавливает необходимые параметры для элемента посредством изменения доступных полей и переключателей в панели справа от рабочей области. Далее пользователь нажимает кнопку «Изменить», и изменения вступают в силу.

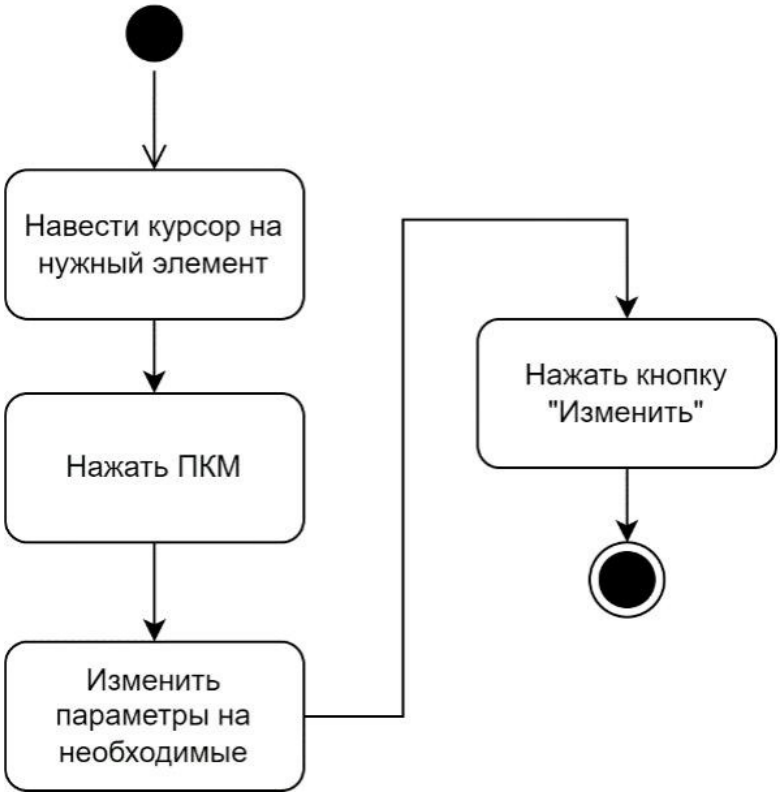

Рисунок 4 - Диаграмма деятельности для сценария «Изменение параметров элемента»
DOI: https://doi.org/10.60797/IRJ.2024.149.110.4

Сценарий «Удаление элементов» заключается в том, чтобы удалить элемент со схемы, пользователь наводит указатель мыши на нужный элемент, затем нажимает правую кнопку мыши. Элемент становится выделенным. Затем пользователь повторно нажимает правую кнопку мыши, и элемент удаляется из рабочей области.

Основным объектом моделирования в рамках данной работы является цифровая схема. Введем сущность «Схема», которая будет включать в себя все элементы, проводники и контакты, размещенные в рабочей области.

В реальных цифровых схемах многие процессы проходят параллельно, например, когда сигнал от одного выходного контакта поступает сразу на несколько входных. Использование методов параллельного программирования позволило бы имитировать такие процессы, однако такой подход связан с излишним усложнением модели, создает проблемы синхронизации доступа к данным, а также ведет к повышению сложности поиска ошибок при отладке программы. В связи с этим был выбран подход пошагового расчета состояния схемы через определенные промежутки времени.

Сущность «Схема» имеет следующий набор атрибутов: список цифровых сигналов, список генераторов, список элементов, список проводов, список измененных входных контактов и список измененных выходных контактов.

Опишем алгоритм расчета схемы. В начале просматривается список логических сигналов. Выход каждого изменившегося цифрового сигнала помещается в список измененных выходных контактов. Аналогичные действия проводятся со списком генераторов. Далее рассматриваются контакты из списка измененных выходных контактов. В списке проводников ищется такой, для которого данный контакт является источником. Если такой не найден, происходит переход к следующему контакту. В противном случае значение этого контакта передается контакту-приемнику (при наличии), и если значение этого контакта изменилось, то он добавляется в список измененных входных контактов. Если к данному проводнику подключены другие проводники, то данное значение передается и их приемникам. Затем рассматриваются списки подключенных к ним проводников и так далее. На следующем шаге текущий контакт удаляется из списка измененных выходных контактов. Если список после этого не пустой, рассматривается первый контакт, иначе переход к следующему этапу расчета.

На следующем этапе рассматривается список измененных входных контактов. Если он не пустой, то выбирается первый. Если элемент-владелец текущего входного контакта еще не рассчитывался на данном этапе, то для него производится расчет значений выходных контактов. Каждый измененный выходной контакт помещается в список измененных выходных контактов. Далее текущий входной контакт удаляется из списка измененных входных контактов. Если список после этого не пустой, этап повторяется, иначе текущая итерация расчета прекращается.

Данный подход позволяет гарантировать, что если между любыми двумя парами элементов находится равное количество других элементов, то одновременно поданные сигналы на входы первых элементов пар повлияют на значения входных и выходных контактов вторых элементов пар за одинаковое число итераций расчета состояний схемы. Кроме того, данный подход за счет не непрерывного пересчета состояний схемы не блокирует работу пользовательского интерфейса.

При взаимодействии с программой пользователь получает информацию о текущем состоянии и структуре схемы посредством восприятия её изображения в рабочей области на экране монитора. Таким образом, важной задачей является отрисовка схемы, в связи с чем вводится понятие отрисовщика схемы.

Отрисовщик схемы – это сущность, основной задачей которой является перерисовка схемы при изменении её состояния, прокрутке в рабочей области, добавлении и удалении новых элементов, изменении их параметров и соединении. Кроме того, он должен отрисовывать временные элементы – элементы и проводники, которые пользователь желает добавить в рабочую область, но ещё не определился с их расположением в её пределах.

К атрибутам отрисовщика схемы относятся временный элемент, временный список точек, последняя позиция мыши, текущий вход, текущий выход, текущий провод, текущий контакт, флаг рисования временных элементов и флаг «Временный элемент – провод», а также схема, которую требуется нарисовать. Временный элемент – это элемент, который пользователь планирует добавить, но пока не выбрал точное место в рабочей области. Временный список точек – это список точек для нового провода. Последняя позиция мыши – это точка в пределах рабочей области, на которую в последний раз указывал курсор мыши. Текущий вход – это входной контакт, являющийся приёмником для рисуемого временного провода. Текущий выход – это выходной контакт, являющийся источником для рисуемого временного провода. Текущий провод – это проводник, на который указывает последняя позиция мыши. Текущий контакт – это контакт, на который указывает последняя позиция мыши. Флаг рисования временных элементов – это флаг, разрешающий или запрещающий отрисовку временных элементов и рисуемых проводов в рабочей области. Флаг «Временный элемент – провод» определяет, отрисовывается в данный момент временный провод или временный элемент. Схема – это схема, которую требуется нарисовать.

Реализация проекта приложения

Для реализации проекта были выбраны язык программирования C#, интегрированная среда разработки Visual Studio и платформа Windows Forms [10].

C# – это объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он сочетает возможности C++ с удобством платформы .NET, обеспечивая безопасное управление памятью и высокую степень безопасности кода. C# активно используется для создания настольных, веб- и мобильных приложений, а также игр и служб. Синтаксис языка, основанный на C++, дополнен многочисленными улучшениями. Язык также предлагает богатую стандартную библиотеку .NET Framework, которая включает функции для работы с файлами, сетями, базами данных и другими задачами.

Windows Forms (WinForms) – это технология, позволяющая создавать графические пользовательские интерфейсы (GUI) в приложениях на C# [11]. WinForms использует визуальный конструктор в Visual Studio, который упрощает процесс создания приложений. Платформа предоставляет множество встроенных элментов управления (контролов), таких как кнопки, текстовые поля, списки и таблицы, а также инструменты для управления их расположением на форме. Контролы генерируют события в ответ на действия пользователя, что позволяет разработчикам обрабатывать эти события для выполнения нужных операций. Кроме того, WinForms поддерживает работу с данными, включая их связывание с контролами и выполнение запросов к базам данных [12].

Разработка приложения начинается с построения главного окна программы. Вначале имеется пустая форма, на которую с помощью конструктора, предоставляемого платформой Windows Forms, добавляются необходимые элементы.

Первым шагом добавляется панель в верхней части формы, в которой будут расположены кнопки «Сохранить», «Открыть» и другие. Для этого выбирается элемент Panel, которому присваивается имя `panelMenu`, высота панели устанавливается на 50, а свойство Dock задается как Top, что привязывает верхнюю границу панели к контейнеру — окну формы. Для визуального отделения панели от других частей формы свойству BorderStyle присваивается значение FixedSingle.

Затем в эту панель добавляются кнопки. Для этого выбирается элемент Button и размещается в панели `panelMenu`. Кнопке присваивается имя `buttonSave`, свойству Text – значение «Сохранить», а свойству Size – значение (100; 50). Аналогично создаются кнопки `buttonOpen` и `buttonStart`, с текстовыми значениями «Открыть» и «Старт» соответственно.

Для создания рабочей области и боковой панели справа используется элемент управления SplitContainer, который состоит из подвижной строки, разделяющей область отображения контейнера на две панели с изменяемыми размерами. Элементу присваивается имя `splitContainerWorkSpace`, а свойство Dock устанавливается на Fill, что привязывает все границы элемента управления к контейнеру – окну формы. Свойству SplitterDistance задается значение 782, при котором ширина правой панели составляет 200 при открытии программы.

Левой и правой панелям задается свойство AutoScroll = true, что управляет автоматическим появлением полосы прокрутки, когда содержимое элемента управления больше его отображаемой области. В левую панель добавляется элемент управления `pictureBox`, предназначенный для отображения рисунков. Он размещается в левом верхнем углу панели, его свойству Size задается значение (2000;2000), а свойству BackColor – значение Windows.

Кнопки для добавления триггеров и некоторых других элементов будут добавлены по мере реализации соответствующих классов. Правая панель элемента `splitContainerWorkSpace` также будет оформлена по мере реализации функционала, связанного с изменением параметров элементов и состояния множества сигналов или триггеров. На текущем этапе вид окна приложения представлен на рисунке 5.

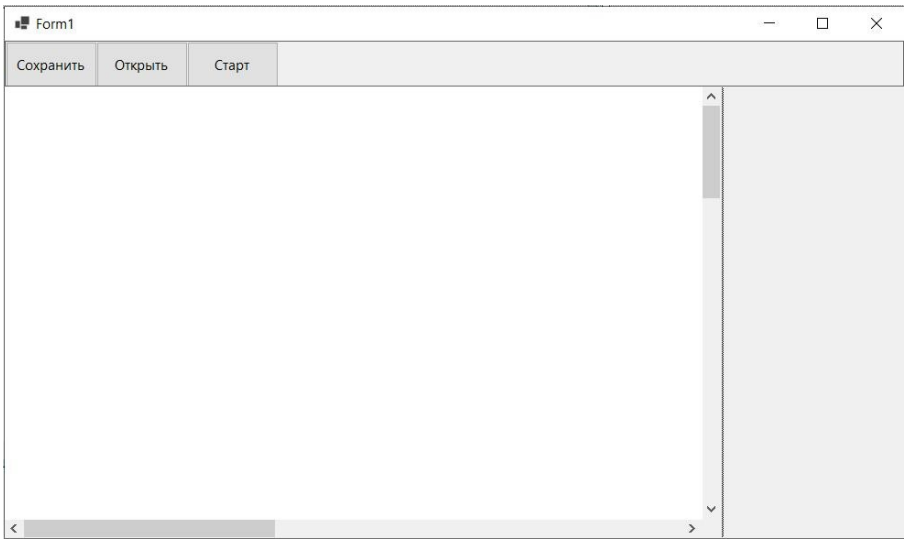

Рисунок 5 - Вид окна приложения на начальном этапе разработки
DOI: https://doi.org/10.60797/IRJ.2024.149.110.5

Далее перечислим интерфейс и классы, который обеспечивают программную реализация проекта приложения:

1. Интерфейс IInOut в C# представляет собой контракт, определяющий набор методов, свойств, событий и индексаторов, которые должны быть реализованы классом или структурой. Он описывает функциональность, которую должен предоставлять класс, но не содержит конкретной реализации. В интерфейс IInOut включены следующие методы и свойства: массив входных контактов элемента (In), массив выходных контактов элемента (Out), строковые массивы имен входных и выходных контактов (InNames и OutNames), строковое значение типа элемента (Type), булево значение инверсии элемента (IsInverted), булево значение изменения состояния контактов (IsChanged), прямоугольник, определяющий положение и размер элемента (Rect), метод вычисления значений выходных контактов (Compute()) и метод перемещения элемента (Move(int x, int y)).

2. Класс Contact: Класс Contact реализует контакты для элементов схемы и содержит следующие свойства: полярность контакта (IsInput), текущее значение (Value), соединение с проводником (IsConnected), область подключения проводника (ContactPlace), координаты точки соединения (ContactPoint) и флаг изменения значения контакта (IsChanged). Конструктор класса принимает параметры: элемент-владелец (Owner), полярность контакта (input) и массив точек, определяющих положение контакта (contactLine). Метод ResetChangedFlag() сбрасывает флаг изменения значения контакта.

3. Класс LogicGate реализует логический вентиль и включает в себя свойства: инверсия вентиля (IsInverted), набор входных и выходных контактов (In и Out), тип вентиля (Type), состояние изменения контактов (IsChanged) и

положение элемента (Rect). Класс реализует интерфейс IInOut и включает метод Compute() для расчета значений выходных контактов. В классе также реализованы функции для логических вентилей (AND, OR, NOT и их комбинации). Конструктор класса принимает параметры: тип вентиля (Type), количество входных контактов (inCnt), инверсию (invert) и положение элемента в рабочей области (Rect).

4. Класс Signal реализует цифровой сигнал и включает в себя свойства: набор входных (In) и выходных контактов (Out), положение элемента (Rect), тип сигнала (Type), состояние изменения контактов (IsChanged) и их инверсию (IsInverted). Класс реализует интерфейс IInOut и включает методы для сброса флага изменения (ResetChangedFlag), вычисления значений выходных контактов (Compute) и инверсии значения сигнала (Invert). Конструктор класса принимает параметр – прямоугольник, определяющий положение сигнала в рабочей области (Rect).

5. Класс Generator реализует генератор тактовых импульсов и включает в себя свойства: набор входных (In) и выходных контактов (Out), положение элемента (Rect), период генератора (T), длительность импульса (tImp) и состояние изменения выходного значения (IsChanged). Класс реализует интерфейс IInOut и включает методы для сброса флага изменения (ResetChangedFlag), вычисления значений выходных контактов (Compute) и обработки времени. Конструктор класса принимает параметры: прямоугольник (Rect), коэффициент заполнения (k) и период генератора в миллисекундах (T).

6. Класс Wire реализует проводник и включает в себя свойства: контакт-источник (Source), контакт-приемник (Receiver), родительский проводник (ParentWire), дочерние проводники (ChildWires), список точек проводника (Points) и флаг выбора проводника (IsSelected). Класс также включает методы для удаления проводника (Delete), удаления и добавления дочерних проводников (RemoveChild и AddChild), установки родительского проводника (SetParent) и присваивания списка точек проводника (SetPoints). Конструкторы класса принимают параметры: родительский проводник (parent), контакт-приемник (receiver) и список точек (Points), либо контакт-источник (source), контакт-приемник (receiver) и список точек (Points).

7. Класс Trigger реализует триггер и включает в себя свойства: набор входных (In) и выходных контактов (Out), имена входных и выходных контактов (InNames и OutNames), положение элемента (Rect), временные значения состояния (tempval_1 и tempval_2) и тип триггера (Type). Класс реализует интерфейс IInOut и включает методы для вычисления значений выходных контактов (Compute). Конструктор класса принимает параметры: прямоугольник (Rect) и тип триггера (RS, D, T или JK).

8. Класс Decoder реализует дешифратор и включает в себя свойства: набор входных (In) и выходных контактов (Out), имена входных и выходных контактов (InNames и OutNames), положение элемента (Rect) и инверсию входов и выходов (IsInverted). Класс реализует интерфейс IInOut и включает метод Compute() для вычисления значений выходных контактов. Конструктор класса принимает параметры: прямоугольник (Rect) и количество адресных входов дешифратора (addressCnt).

9. Класс Mux реализует мультиплексор и включает в себя свойства: набор входных (In) и выходных контактов (Out), имена входных и выходных контактов (InNames и OutNames), положение элемента (Rect) и количество адресных входов (addressCnt). Класс реализует интерфейс IInOut и включает метод Compute() для вычисления значений выходных контактов. Конструктор класса принимает параметры: прямоугольник (Rect) и количество адресных входов мультиплексора (addressCnt).

10. Класс TempElement содержит логику создания временных элементов и включает в себя свойства: тип элемента (Type), количество входов (InCount) и выходов (OutCount), положение элемента (Rect). Класс также включает методы для изменения типа элемента (ChangeType), количества входов (ChangeInCount), положения элемента (ChangePosition) и создания нового элемента (CreateLogical).

11. Класс Schema реализует цифровую схему и включает в себя свойства: список всех элементов схемы (logicals), список проводников (wires), список измененных контактов (changedOutputContacts и changedInputContacts), список генераторов (Generators) и состояние схемы (isFirstRun). Класс также включает методы для расчета состояния схемы (Calculate), добавления и удаления элементов и проводников (Add и Remove), а также получения индекса проводника, к которому подключен контакт (GetWireIndex).

12. Класс SchemaDrawer предназначен для отрисовки схемы. Список полей класса включает: схему для отрисовки (schema), перо для отрисовки (pen и tempPen), временный элемент (temp), список точек временного провода (tempWirePoints), координаты последнего расположения курсора (lastMouseLock), активный контакт (activeContact), приемник и источник создаваемого провода (currentIn и currentOut), текущий проводник (currentWire), ранее выбранный проводник (prevSelectedWire), выбранный элемент (selectedElement), булевые значения для рисования временных элементов (isTempDrawable и tempIsWire). Класс также имеет публичное событие OnSelected, вызываемое при выделении или снятии выделения с какого-либо элемента.

13. Класс Form1 представляет окно, которое составляет пользовательский интерфейс приложения. Он включает элементы правой панели для изменения параметров выделенных элементов, множественного изменения состояний входных сигналов и триггеров, отслеживания выходных сигналов. На рисунке 6 представлен вид окна программы на данном этапе разработки. Элементы правой панели предназначены для изменения параметров выделенных элементов, множественного изменения состояний входных сигналов и триггеров, отслеживания выходных сигналов.

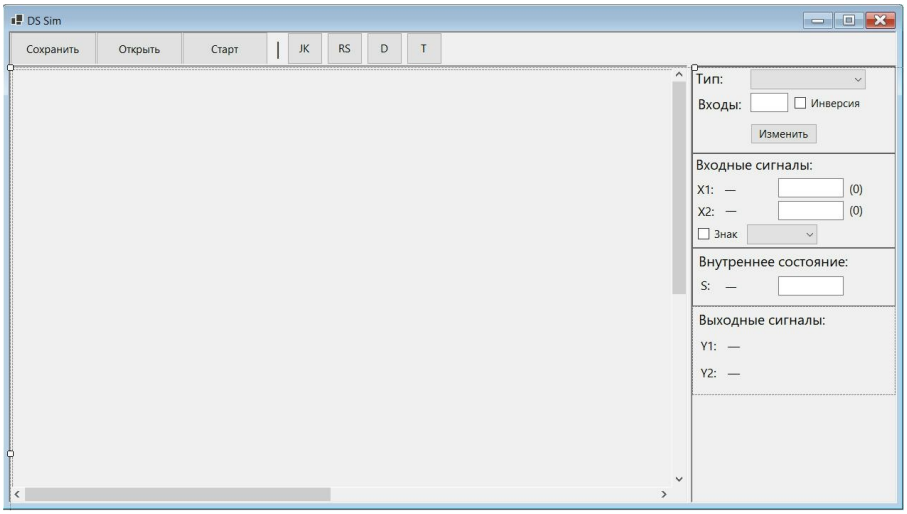

Рисунок 6 - Окно программы
DOI: https://doi.org/10.60797/IRJ.2024.149.110.6

Тестирование

Метод тестирования заключается в том, чтобы с помощью клавиш на клавиатуре и кнопок на форме добавлять различные элементы с разными параметрами. Контроль за добавлением элементов осуществляется путем отслеживания содержимого списка logicals – поля класса Schema.

Добавим на схему 18 элементов: вентили в различных конфигурациях, сигнал, генератор, дешифраторы, мультиплексоры, триггеры. Состояние списка logicals и схемы представлены на рисунке 7.

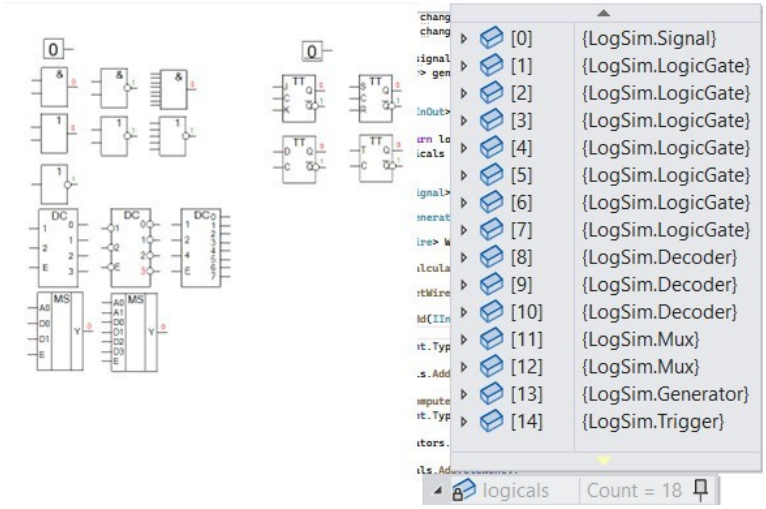

Рисунок 7 - Схема и состояние списка logicals
DOI: https://doi.org/10.60797/IRJ.2024.149.110.7

Как видно на рисунке 7, размер списка равен 18 – количеству добавленных элементов, кроме того, каждый из элементов добавлен с верным количеством входов, имеет инвертированные контакты, если такие необходимы.

Для проверки корректности соединения элементов и проверка корректности их работы, соединим несколько сигналов со входами элементов как на рисунке 8.

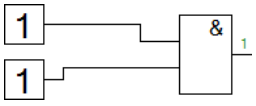

Рисунок 8 - Соединение элементов
DOI: https://doi.org/10.60797/IRJ.2024.149.110.8

При двойном клике по сигналу его состояние меняется на противоположное, при этом происходит корректное изменение выхода элемента И, что видно на рисунке 9.

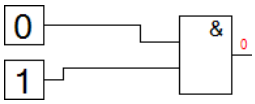

Рисунок 9 - Изменение сигнала
DOI: https://doi.org/10.60797/IRJ.2024.149.110.9

Подобным образом были проверены логические вентили с различными наборами входных сигналов и количеством входов. Все логические вентили работают корректно.

Соединим вентили с сигналами, как на рисунке 10 и изменим входные сигналы.

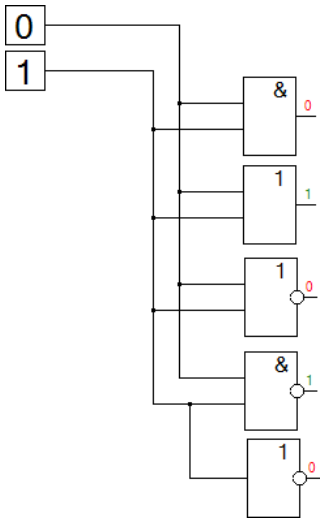

Рисунок 10 - Подключение нескольких приемников к проводнику
DOI: https://doi.org/10.60797/IRJ.2024.149.110.10

При изменении состояния сигналов вентили работают корректно, следовательно, передачи сигналов от источника до приемника и от проводника к проводнику работают корректно.

На рисунке 11 показана работа дешифратора.

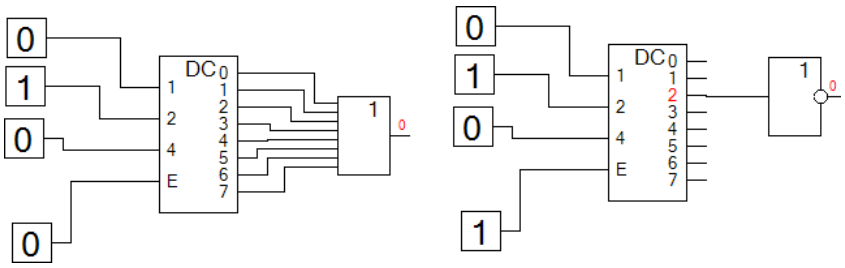

Рисунок 11 - Работа дешифратора
DOI: https://doi.org/10.60797/IRJ.2024.149.110.11

Демонстрация работы инверсного дешифратора представлена на рисунке 12.

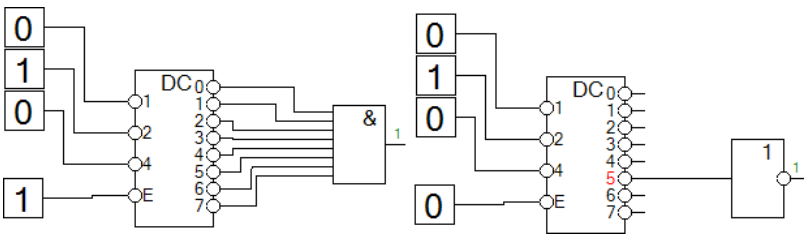

Рисунок 12 - Работа инверсного дешифратора
DOI: https://doi.org/10.60797/IRJ.2024.149.110.12

Демонстрация работы мультиплексора представлена на рисунке 13.

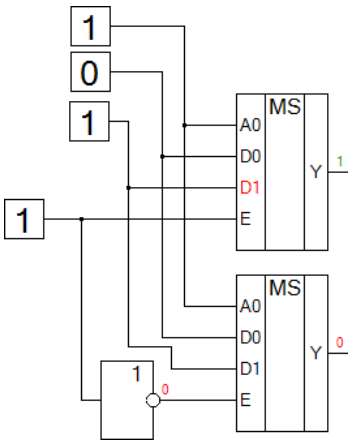

Рисунок 13 - Работа мультиплексора
DOI: https://doi.org/10.60797/IRJ.2024.149.110.13

На рисунке 14 показана работа триггеров.

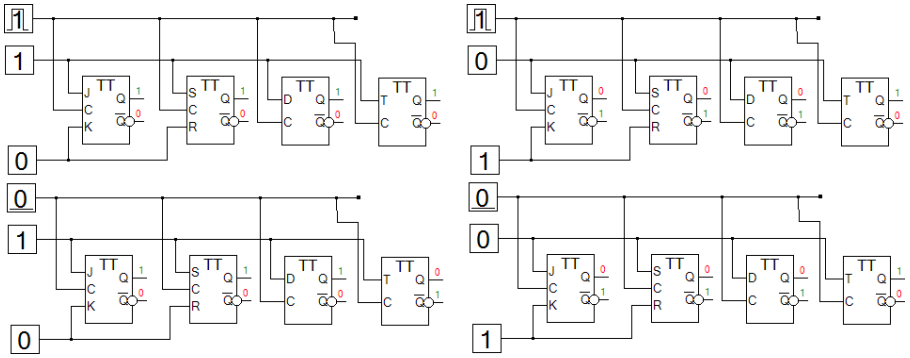

Рисунок 14 - Работа триггеров
DOI: https://doi.org/10.60797/IRJ.2024.149.110.14

По итогам испытаний все элементы работают корректно.

Для тестирования удаления проводников и элементов будем следить за списками проводников и элементов, а также изменять состояния сигналов после их разъединения с элементом, чтобы убедится, что они более не влияют на отключенный элемент.

Результаты испытаний представлены на рисунке 15.

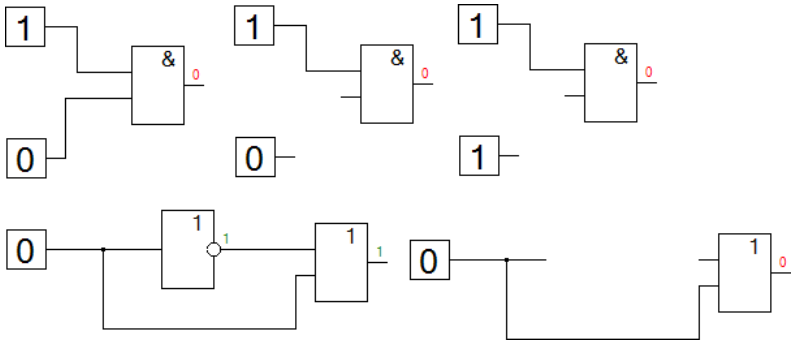

Рисунок 15 - Удаление элементов и проводников
DOI: https://doi.org/10.60797/IRJ.2024.149.110.15

Наблюдение за списками проводников и элементов подтвердило, что удаляются именно те объекты, которые планировалось удалить. Корректность удаления проводника показана на рисунке 16.

Рисунок 16 - Изменение списка проводников при удалении проводника
DOI: https://doi.org/10.60797/IRJ.2024.149.110.16

Демонстрация работы группового изменения входных сигналов показана на рисунке 17.

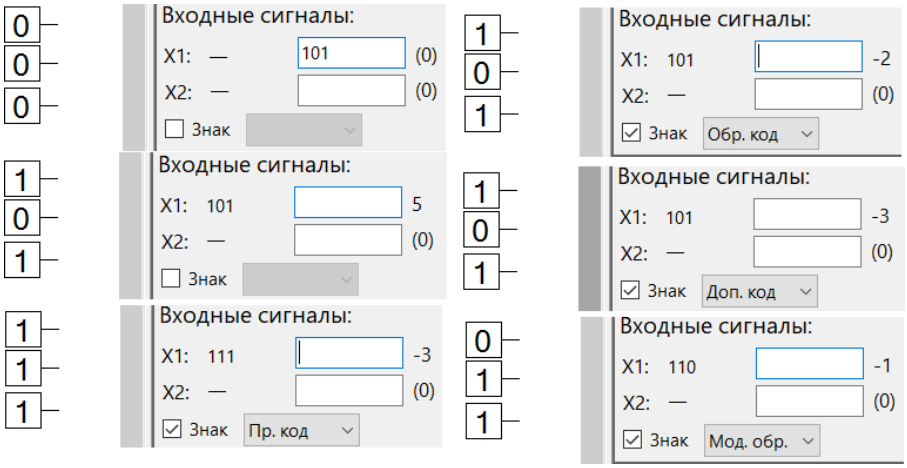

Рисунок 17 - Групповое изменение входных сигналов
DOI: https://doi.org/10.60797/IRJ.2024.149.110.17

Отметим также корректную работу отображения слева от поля ввода, закодированного битами выходных значений сигналов числа. Старший бит находится снизу. Кроме того, отображаются числовые значения сигналов, которые образуют прямой, обратный, дополнительный и модифицированный обратный коды.

Прямой код – старший бит кода равен нулю, остальные биты представляют двоичное представление числа. Дополнительный код получается из прямого путем инверсии с последующим добавлением единицы.

Демонстрация корректности отображения выходных сигналов представлена на рисунке 18.

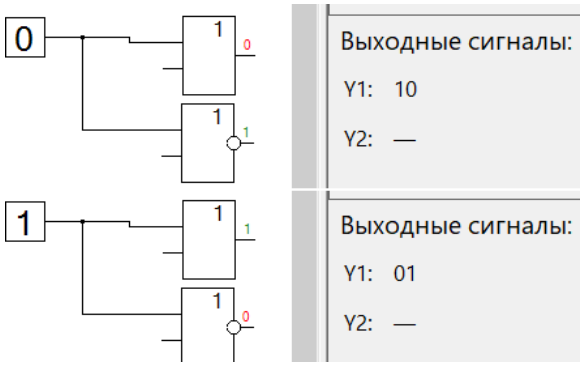

Рисунок 18 - Отображение значений выходных сигналов
DOI: https://doi.org/10.60797/IRJ.2024.149.110.18

Заключение

Основные задачи данной работы, включающие исследование предметной области, создание проектной части, разработку приложения и тестирование, успешно выполнены.

На основе анализа предметной области были проанализированы существующие решения, выявлены их преимущества и недостатки, а также рассмотрены теоретические сведения о цифровых схемах. В ходе работы над проектной частью были выделены основные сущности и их атрибуты, а затем составлены их модели. Кроме того, были разработаны основные сценарии взаимодействия пользователя с приложением.

В процессе разработки модели были реализованы на языке программирования C#. В ходе тестирования проверялись на корректность различные компоненты разработанного приложения, и ошибок выявлено не было.

Таким образом, результатом работы является создание приложения для моделирования цифровых схем, совмещающее простоту использования и функциональность, чего не достигалось в рассмотренных в рамках работы аналогах. Данное приложение может найти применение в образовательном процессе по направлению подготовки 09.03.01 «Информатика и вычислительная техника».

Дальнейшее развитие приложения для моделирования цифровых схем может включать в себя несколько ключевых направлений. Одним из них является расширение набора компонентов, что позволит включить дополнительные логические элементы, увеличивая тем самым возможности пользователей и позволяя создавать более сложные схемы. Оптимизация пользовательского интерфейса является важным направлением для улучшения взаимодействия с приложением. Введение инструментов для упрощения работы с большими схемами, таких как автоматическое выравнивание элементов, удобная навигация и зуммирование, сделает процесс проектирования более интуитивным. Поддержка симуляции и отладки, включая функционал для пошаговой симуляции работы схемы, возможности вставки точек останова и анализа промежуточных сигналов, поможет пользователям лучше понять работу схем и отладить их. Интеграция с обучающими платформами и курсами по цифровой электронике сделает приложение полезным в образовательных целях, предоставляя студентам интерактивные задачи и тесты. Добавление мультиязыковой поддержки расширит аудиторию приложения, сделав его доступным для пользователей по всему миру. Введение библиотек и шаблонов для типовых задач ускорит процесс проектирования и позволит пользователям использовать проверенные решения. Совместимость с аппаратными средствами, такими как физические устройства и микроконтроллеры [13], [14], позволит пользователям тестировать свои схемы в реальных условиях. Внедрение облачных сервисов для хранения и обмена проектами обеспечит возможность совместной работы над схемами, делиться своими проектами и получать обратную связь. Разработка мобильных и веб-версий приложения [15], [16] позволит пользователям работать с приложением в любом месте и в любое время, обеспечивая удобный доступ с различных устройств. Эти направления развития помогут сделать приложение более мощным, удобным и востребованным инструментом для моделирования цифровых схем, удовлетворяющим потребности как начинающих, так и опытных пользователей.

Конфликт интересов

Не указан.

Рецензия

Белашова Е.С., Казанский национальный исследовательский технический университет им. А.Н. Туполева – КАИ, Казань, Российская Федерация
DOI: https://doi.org/10.60797/IRJ.2024.149.110.19

Conflict of Interest

None declared.

Review

Belashova E.S., Kazan National Research Technical University named after A.N. Tupolev – KAI, Kazan, Russian Federation
DOI: https://doi.org/10.60797/IRJ.2024.149.110.19

Список литературы / References

1. Колесникова Т. Инструменты анализа схем электрических принципиальных в программной среде NI Multisim 12.0 / Т. Колесникова // Компоненты и технологии. — 2015. — № 1(162). — С. 113-120.

2. Найденко Е. В. Использование программной среды NI MULTISIM при подготовке студентов направлений "электромеханика" и "электротехника" / Е. В. Найденко, Е. Ю. Маевская // Электротехнические и компьютерные системы. — 2017. — № 24(100). — С. 164–168.

3. Комарова Э. П. Система схемотехнического моделирования Micro-Cap 6.0 / Э. П. Комарова [и др.]; науч. ред. Г. В. Макаров. — Второе изд., перераб. и доп. — Воронеж : Воронеж. гос. техн. ун-т, 2004. — 21 с.

4. Амелина М. А. Программа схемотехнического моделирования Micro-Cap 8 / М. А. Амелина, С. А. Амелин. — Москва : Горячая линия-Телеком, 2007. — 464 с. — ISBN 978-5-93517-339-5.

5. Иванов В. И. Схемотехника ЭВМ. Проектирование цифровых узлов на интегральных схемах средней степени интеграции : учебное пособие / В. И. Иванов. — Красноярск : КГТУ, 2006. — 190 с.

6. Строганов А. В. Основы проектирования цифровых интегральных схем : учебное пособие / А. В. Строганов, А. В. Строгонов. — Воронеж : Воронежский гос. технический ун-т, 2008. — 184 с.

7. Райхлин В. А. Конструктивное моделирование процессов синтеза / В. А. Райхлин, И. С. Вершинин, Р. К. Классен [и др.]. — Казань : Фэн, 2020. — 248 с.

8. Мышляева И. М. Цифровая схемотехника / И. М. Мышляева. — Москва : Academia, 2005.

9. Ашихмин А. С. Цифровая схемотехника. Шаг за шагом : обработка цифровых данных, основные узлы цифровых устройств, практикум проектирования на базе ПЛИС, САПР Quartus и фирмы Altera, микросхемы FPGA Cyclone III / А. С. Ашихмин. — Москва : Диалог-МИФИ, 2008.

10. Мурлин А. Г. Программирование с использованием компонентов Windows Forms : учебное пособие / А. Г. Мурлин, В. А. Мурлина, М. В. Янаева. — Краснодар : Издательский Дом - Юг, 2012. — 126 с.

11. Гибадуллин Р. Ф. Параллельное программирование на языках C/C++ и C# / Р. Ф. Гибадуллин, Е. С. Белашова. — Казань : Редакционно-издательский центр "Школа", 2021. — 104 с.

12. Гибадуллин Р. Ф. Система баз данных картографии с ассоциативной защитой : специальность 05.13.19 "Методы и системы защиты информации, информационная безопасность" : диссертация на соискание ученой степени кандидата технических наук / Гибадуллин Руслан Фаршатович. — Казань, 2011. — 117 с.

13. Микропроцессорные, аналоговые и цифровые системы : проектирование и схемотехника, теория и вопросы применения : материалы 5-й Международной научно-практической конференции, Новочеркасск, 18 февраля 2005 года. — Новочеркасск : Южно-Российский государственный политехнический университет (НПИ) имени М.И. Платова, 2005. — 55 с.

14. Гибадуллин Р. Ф. Организация защищенной передачи данных в сенсорной сети на базе микроконтроллеров AVR / Р. Ф. Гибадуллин // Кибернетика и программирование. — 2018. — № 6. — С. 80–86. DOI: 10.25136/2306-4196.2018.6.24048.

15. Гергерт А. В. Современные облачные архитектуры на рынке веб-приложений / А. В. Гергерт, А. П. Загородников // Энергосбережение и инновационные технологии в топливно-энергетическом комплексе : Материалы Национальной с международным участием научно-практической конференции студентов, аспирантов, ученых и специалистов, посвященной 20-летию создания кафедры электроэнергетики: в 2-х томах, Тюмень, 18–20 декабря 2019 года. — Тюмень : Тюменский индустриальный университет, 2019. — С. 14–17.

16. Gibadullin R. F. Mobile Application for Neural Network Analysis of Human Functional State / R. F. Gibadullin, R. R. Zakirov // Lecture Notes in Electrical Engineering. — 2021. — Vol. 729. — P. 745–755. DOI: 10.1007/978-3-030-71119-1_73.

Список литературы на английском языке / References in English

1. Kolesnikova T. Instrumenty analiza shem jelektricheskih principial'nyh v programmnoj srede NI Multisim 12.0 [Tools for analyzing electrical circuit diagrams in the NI Multisim 12.0 software environment] / T. Kolesnikova // Komponenty i tehnologii [Components and Technologies]. — 2015. — № 1(162). — P. 113–120. [in Russian]

2. Naidenko E. V. Ispol'zovanie programmnoj sredy NI MULTISIM pri podgotovke studentov napravlenij "jelektromehanika" i "jelektrotehnika" [The use of the NI MULTISIM software environment in the preparation of students in the fields of "electromechanics" and "electrical engineering"] / E. V. Naidenko, E. Y. Mayevskaya // Jelektrotehnicheskie i komp'juternye sistemy [Electrical engineering and computer systems]. — 2017. — № 24(100). — P. 164–168. [in Russian]

3. Komarova E. P. Sistema shemotehnicheskogo modelirovanija Micro-Cap 6.0 [Micro-Cap 6.0 circuit modeling system] / E. P. Komarova [et al.]; scientific ed. by G. V. Makarov. — Second edition, revised. and add. — Voronezh : Voronezh State Technical University. Univ., 2004. — 21 p. [in Russian]

4. Amelina M. A. Programma shemotehnicheskogo modelirovanija Micro-Cap 8 [Micro-Cap 8 circuit modeling program] / M. A. Amelina, S. A. Amelin. — Moscow : Hotline-Telecom, 2007. — 464 p. — ISBN 978-5-93517-339-5. [in Russian]

5. Ivanov V. I. Shemotehnika JeVM. Proektirovanie cifrovyh uzlov na integral'nyh shemah srednej stepeni integracii [Computer circuitry. Designing digital nodes on integrated circuits of medium degree of integration] : a textbook / V. I. Ivanov. — Krasnoyarsk : KSTU, 2006. — 190 p. [in Russian]

6. Stroganov A. V. Osnovy proektirovanija cifrovyh integral'nyh shem [Fundamentals of designing digital integrated circuits] : a textbook / A. V. Stroganov, A. V. Strogonov. — Voronezh : Voronezh State Technical University, 2008. — 184 p. [in Russian]

7. Raikhlin V. A. Konstruktivnoe modelirovanie processov sinteza [Constructive modeling of synthesis processes] / V. A. Raikhlin, I. S. Vershinin, R. K. Klassen [et al.]. — Kazan : Feng, 2020. — 248 p. [in Russian]

8. Myshlyaeva I. M. Cifrovaja shemotehnika [Digital circuitry] / I. M. Myshlyaeva. — Moscow : Academia, 2005. [in Russian]

9. Ashikhmin A. S. Cifrovaja shemotehnika. Shag za shagom : obrabotka cifrovyh dannyh, osnovnye uzly cifrovyh ustrojstv, praktikum proektirovanija na baze PLIS, SAPR Quartus i firmy Altera, mikroshemy FPGA Cyclone III [Digital circuitry. Step by step : digital data processing, the main nodes of digital devices, a design workshop based on FPGA, Quartus CAD and Altera, FPGA Cyclone III chips] / A. S. Ashikhmin. — Moscow : Dialog-MEPhI, 2008. [in Russian]

10. Murlin A. G. Programmirovanie s ispol'zovaniem komponentov Windows Forms [Programming using Windows Forms components] : a textbook / A. G. Murlin, V. A. Murlina, M. V. Yanaeva. — Krasnodar : Publishing House - Yug, 2012. — 126 p. [in Russian]

11. Gibadullin R. F. Parallel'noe programmirovanie na jazykah S/C++ i C# [Parallel programming in C/C++ and C# languages] / R. F. Gibadullin, E. S. Belashova. — Kazan : Editorial and publishing center "School", 2021. — 104 p. [in Russian]

12. Gibadullin R. F. Sistema baz dannyh kartografii s associativnoj zashhitoj [Database system of cartography with associative protection] : specialty 05.13.19 "Methods and systems of information protection, information security" : dissertation for the degree of Candidate of Technical Sciences / Gibadullin Ruslan Farshatovich. — Kazan, 2011. — 117 p. [in Russian]

13. Mikroprocessornye, analogovye i cifrovye sistemy : proektirovanie i shemotehnika, teorija i voprosy primenenija [Microprocessor, analog and digital systems : design and circuit engineering, theory and application issues] : proceedings of the 5th International Scientific and Practical Conference, Novocherkassk, February 18, 2005. — Novocherkassk : South Russian State Polytechnic University (NPI) named after M.I. Platov, 2005. — 55 p. [in Russian]

14. Gibadullin R. F. Organizacija zashhishhennoj peredachi dannyh v sensornoj seti na baze mikrokontrollerov [Organization of secure data transmission in a sensor network based on AVR microcontrollers] / R. F. Gibadullin // Kibernetika i programmirovanie [Cybernetics and Programming]. — 2018. — № 6. — P. 80–86. DOI: 10.25136/2306-4196.2018.6.24048. [in Russian]

15. Gergert A. V. Sovremennye oblachnye arhitektury na rynke veb-prilozhenij [Modern cloud architectures in the web application market] / A. V. Gergert, A. P. Zagorodnikov // Jenergosberezhenie i innovacionnye tehnologii v toplivno-jenergeticheskom komplekse [Energy saving and innovative technologies in the fuel and energy complex] : Materials of the National Scientific and Practical Conference of students, postgraduates, scientists and specialists with international participation, dedicated to the 20th anniversary of the creation of the Department of Electric Power Engineering: in 2 Tomakh, Tyumen, December 18-20, 2019. — Tyumen : Tyumen Industrial University, 2019. — P. 14–17. [in Russian]

16. Gibadullin R. F. Mobile Application for Neural Network Analysis of Human Functional State / R. F. Gibadullin, R. R. Zakirov // Lecture Notes in Electrical Engineering. — 2021. — Vol. 729. — P. 745–755. DOI: 10.1007/978-3-030-71119-1_73.