# Содержание

[Введение 2](#_Toc289090749)

[1. Обзор методов логического моделирования 3](#_Toc289090750)

[2. Описание используемого алгоритма логического моделирования 5](#_Toc289090751)

[3. Разработка программного обеспечения 7](#_Toc289090752)

[3.1 Описание разработанных типов данных и процедур 7](#_Toc289090753)

[3.2 Описание возможностей программы 12](#_Toc289090754)

[Заключение 20](#_Toc289090755)

[Список литературы 21](#_Toc289090756)

# *Введение*

Широкое распространение радиоэлектронных устройств с применением цифровой обработки сигналов обуславливает повышенный интерес к вопросам диагностирования их технического состояния.

Одной из разновидностей диагностирования цифровых узлов и блоков является тестовое диагностирование, применение которого на этапе проектирования и изготовления цифровых узлов позволяет определить правильность их функционирования. При разработке тестовой диагностики возникает сложность в определении эталонных реакций при тестировании существующих схем и оптимального числа контрольных точек для снятия выходной реакции диагностируемой цифровой схемы. Это можно сделать либо создавая прототип разрабатываемого цифрового устройства и проводя его диагностику аппаратурными методами, либо осуществляя моделирование на ЭВМ как цифрового устройства, так и процесса диагностики. Наиболее рациональным является второй подход, который предполагает создание автоматизированных систем, позволяющих производить диагностику цифровых схем на стадии проектирования. Одними из таких систем являются системы логического моделирования.

Цель логического моделирования состоит в том, чтобы выполнить функцию проектируемой схемы без её физической реализации. Проверка на правильность моделирования может быть различной в зависимости от уровня представления цифровой схемы в ЭВМ. Если, например, осуществляется проверка только значений логической функции на выходе схемы, то достаточно представить схему на уровне логических элементов. Для того чтобы проверить состояния сигналов в схеме, необходимо точно описать задержки срабатывания всех элементов в условиях синхронизации.

В данной работе рассматривается логическое моделирование схем с помощью событийного алгоритма (ЛИД-модель элементов).

# *1. Обзор методов логического моделирования*

В отношении асинхронных моделей возможны два метода логического моделирова­ния — пошаговый (инкрементный) и событийный [2].

В пошаговом методе время дискретизируется и вычисления по выражени­ям модели выполняются в дискретные моменты времени t0, t1, t2,… и т. д. Шаг дискретизации ограничен сверху значением допустимой погрешности определе­ния задержек и потому оказывается довольно малым, а время анализа — значи­тельным.

Для сокращения времени анализа используют событийный метод. В этом методе событием называют изменение любой переменной модели. Событий­ное моделирование основано на следующем правиле: обращение к модели ло­гического элемента происходит только в том случае, если на входах этого эле­мента произошло событие. В сложных логических схемах на каждом такте синхронизации обычно происходит переключение всего лишь 2 ... 3 % логичес­ких элементов, и соответственно в событийном методе в несколько раз умень­шаются вычислительные затраты по сравнению с пошаговым моделировани­ем.

Методы анализа синхронных моделей представляют собой методы реше­ния систем логических уравнений. К этим методам относятся метод простых итераций и метод Зейделя, которые аналогичны одноименным методам решения систем алгебраических уравнений в непрерывной математике.

При использовании метода простых итераций в исходном состоянии задают начальные (можно произвольные) значения промежуточных и выход­ных переменных. Новое состояние должно соответствовать указанным в таблице изменившимся значениям вход­ных сигналов. Вычисления заканчиваются, если на очередной итера­ции изменений переменных нет.

Согласно методу простых итераций, в правые части уравнений модели на каждой итерации подставляют значения переменных, полученные на предыду­щей итерации. В отличие от этого в методе Зейделя, если у некоторой пере­менной обновлено значение на текущей итерации, именно его и используют в дальнейших вычислениях уже на текущей итерации. Метод Зейделя позволяет сократить число итераций, но для этого нужно предварительно упорядочить уравнения модели так, чтобы последовательность вычислений соответствова­ла последовательности прохождения сигналов по схеме. Такое упорядочение выполняют с помощью ранжирования.

Ранжирование заключается в присвоении элементам и переменным моде­ли значений рангов в соответствии со следующими правилами: 1) в схеме раз­рываются все контуры обратной связи, что приводит к появлению дополни­тельных входов схемы (псевдовходов); 2) все внешние переменные (в том числе на псевдовходах) получают ранг 0; 3) элемент и его выходные переменные получают ранг к, если у элемента все входы проранжированы и старший среди рангов входов равен к - 1.

Для сокращения объема вычислений в синхронном моделировании возмож­но использование событийного подхода. Обращение к модели эле­мента происходит, только если на его входах произошло событие.

# *2. Описание используемого алгоритма логического моделирования*

В данной работе для реализации системы логического моделирования использовался событийный алгоритм моделирования логических схем (ЛИД-модель элементов).

Основным свойством событийного алгоритма является то, что на каждом шаге выполняется моделирование работы не всего множества элементов схемы, а только тех элементов, у которых изменился хотя бы один входной сигнал и, следовательно, может измениться выходной сигнал. Моделирование, при котором используется модель, не учитывающая временных характеристик, называется аналитической статической (Л-модель). Модель, любой элемент которой можно представить функциональным и динамическим блоками, учитывает временные характеристики и называется логико-динамической (ЛД-модель). В данной работе используется ЛИД-модель элементов, которая также учитывает инерционные свойства элементов.

Реализация событийного алгоритма требует наличия Таблицы Будущих Событий (ТБС), Таблицы Текущих Событий (ТТС) и счётчика системного времени (ССВ). В ТБС заносятся номера элементов, на входах которых изменились сигналы. На каждом шаге элементу ставится момент времени Т возможного изменения сигнала на выходе элемента, который определяется суммой ССВ, динамической и инерционной задержек элемента. Таблица Текущих Событий (ТТС) содержит номера элементов, для которых выполняется расчёт выходных сигналов на данном шаге.

Начальное значение ССВ равняется 0. Затем подаются нулевые сигналы, и выполняется моделирование схемы по алгоритму простой итерации. После этого подаётся набор входных сигналов, и в ТБС заносятся номера элементов, на которые поступили сигналы. На каждом следующем шаге из ТБС в ТТС переносятся номера тех элементов, Т которых минимальное, и выполняется расчёт выходных сигналов всех элементов из ТТС. Если выходной сигнал элемента изменил своё значение, то в ТБС заносятся номера всех элементов, на входы которых подаётся данный сигнал.

Итерации для заданного набора оканчиваются, когда ТБС пуста, либо ССВ превысил значение ССВk + Tmaх. Tmax равняется задержке самой длинной цепочки элементов.

Блок-схема использованного алгоритма приведена в Приложении А.

# *3. Разработка программного обеспечения*

## *3.1 Описание разработанных типов данных и процедур*

Для выполнения технического задания были разработаны следующие типы данных и процедуры:

* Пакет zak.adcs - содержит главный класс программы и классы для сохранения настроек программы и загрузки изображений.
  + Класс Program – главный класс программы.
  + Класс Configuration – класс для сохранения настроек программы посредством сериализации в XML.
  + Класс ImageIconLoader – класс для загрузки изображений из jar-дистрибутива.
* Пакет zak.adcs.gui – содержит классы для отображения визуальных окон программы.
  + Класс MainFrame – класс для представления главного окна программы и с помощью элементов интерфейса предоставляет доступ к основным функциям программы.
  + Класс InternalFrame – класс для предоставления интерфейса для работы со схемой и моделирования, а также для возможности одновременной работы с множеством схем.
  + Класс SchemeDialog – класс, представляющий диалоговое окно для задания настроек схемы.
* Пакет zak.adcs.logicsheme.elements – содержит классы и интерфейсы для представления и работы с моделями логических элементов.
  + Интерфейс VisualElement – интерфейс элемента в редакторе схемы.
    - public void draw(Graphics2D g2) – метод для отрисовки элемента в редакторе схем.
    - public void move(int x, int y) – метод для перемещения элемента в редакторе схем.
    - public void setSelected(boolean selected) – метод для выделения/отмены выделения элемента в редакторе схем.
    - public boolean contains(Point p) – метод для проверки принадлежности точки области отрисовки элемента.
    - public boolean isOverlaped(Rectangle r) – метод, чтобы проверить не перекрывает ли прямоугольник область отрисовки элемента.
  + Абстрактный класс ClosedElement – класс для представления абстракции элемента с замкнутым контуром в редакторе схем.
  + Интерфейс Enterable – интерфейс элемента, который имеет хотя бы один вход.
    - public boolean isInEnter(Point point) – метод для проверки, находится ли точка в входе в области отрисовки элемента.
    - public Point getEnterPoint(Point point) throws ConnectionAlreadyExistException – метод для получения точки входа в области отрисовки элемента.
    - public String getName() – метод для получения имени элемента.
    - public void addEnterConnectionName(String name, Point point) – метод для для подсоединения выхода другого элемента к входу данного.
  + Интерфейс Exitable – интерфейс элемента, который имеет выход.
    - public boolean isInExit(Point point) – метод для проверки, находится ли точка в выходе в области отрисовки элемента.
    - public Point getExitPoint(Point point) throws ConnectionAlreadyExistException – метод для получения точки выхода в области отрисовки элемента.
    - public String getName() – метод для получения имени элемента.
    - public void addExitConnectionName(String name, Point point) – метод для подсоединения входа другого элемента к выходу данного.
  + Интерфейс LIDElement – интерфейс логического элемента, который имеет динамическую и инерционную задержки.
    - public int getInertiaDelay() – метод для получения инерционной задержки логического элемента.
    - public int getDynamicDelay() – метод для получения динамической задержки логического элемента.
    - public void setInertiaDelay(int inertiaDelay) – метод для установки инерционной задержки логического элемента.
    - public void setDynamicDelay(int dynamicDelay) – метод для установки динамической задержки логического элемента.
  + Класс ConnectionElement – класс для представления соединения между элемента в редакторе схем.
  + Класс InElement – класс для представления входа схемы.
  + Класс OutElement – класс для представления выхода схемы.
  + Абстрактный класс LogicElement – класс для представления логического элемента.
  + Класс AndElement – класс для представления логического элемента И.
  + Класс OrElement – класс для представления логического элемента ИЛИ.
  + Класс NandElement – класс для представления логического элемента И-НЕ.
  + Класс NorElement – класс для представления логического элемента ИЛИ-НЕ.
  + Класс XorElement – класс для представления логического элемента Исключительное ИЛИ.
  + Класс ConnectionAlreadyExistException – класс для представления исключения при попытке добавить соединения к уже занятому входу/выходу.
* Пакет zak.adcs.logicscheme.editor – содержит классы для отображения редактора схем, представления и сохранения/восстановления схемы в памяти.
  + Класс SchemeEditor – класс для отображения рабочей области редактора схем и элементов управления редактором.
  + Класс SchemePanel – класс для отображения рабочей области редактора схем, а также обработки действий пользователя.
    - Внутренний класс MouseHandler – класс для обработки событий мыши в рабочей области редактора схем.
  + Класс IOElementDialog – класс для отображения диалогового окна с параметрами элемента входа/выхода схемы.
  + Класс LIDElementDialog – класс для отображения диалогового окна с параметрами логического элемента.
  + Класс SchemeModel – класс для представления модели схемы в памяти, а также для ее сохранения/восстановления.
* Пакет zak.adcs.logicscheme.modelling – содержит классы для моделирования логических схем и вывода результатов в табличном виде и в виде временной диаграммы.
  + Класс SchemeTableModel – класс для представления матрицы связности элементов схемы.
  + Класс ModellingPanel – класс для отображения результатов моделирования и элементов интерфейса для управления моделированием.
  + Класс TimeDiagramPanel – класс для отображения результатов моделирования в виде временной диаграммы.
  + Класс LogicFunctionsWorker – класс для моделирования логических элементов схемы.
    - Метод public static char getValue(LogicFunction function, String inputSet) – метод для получения значения выхода логического элемента при заданных значениях входных сигналов.
  + Класс ModellingModel – класс для моделирования схем.
    - Метод public boolean isBusy() – метод, определяющий завершено ли моделирование схемы.
    - Метод public void step(String inputSet, String startSet) – метод для моделирования схемы при новом входном наборе с установочным набором на один интервал времени.
    - Метод public void step() – метод для моделирования схемы на один интервал времени.
    - Метод public void step(String inputSet, boolean newSet) – метод для моделирования схемы при новом входном наборе без установочного набора на один интервал времени.
    - Метод public void modelling(String inputSet, String startSet) – метод для моделирования схемы.
    - Метод public void clear() – метод для завершения моделирования и очистки результатов.
    - Приватный метод private String[] getStartCondition(String startSet) – метод для получения начального состояния схемы путем моделирования схемы методом простой итерации.
    - Приватный метод private boolean hasFutureEvents() – метод для проверки наличия будущих событий схемы.

## *3.2 Описание возможностей программы*

При запуске программы на экране появляется главное окно программы, изображенное на рисунке 3.1.

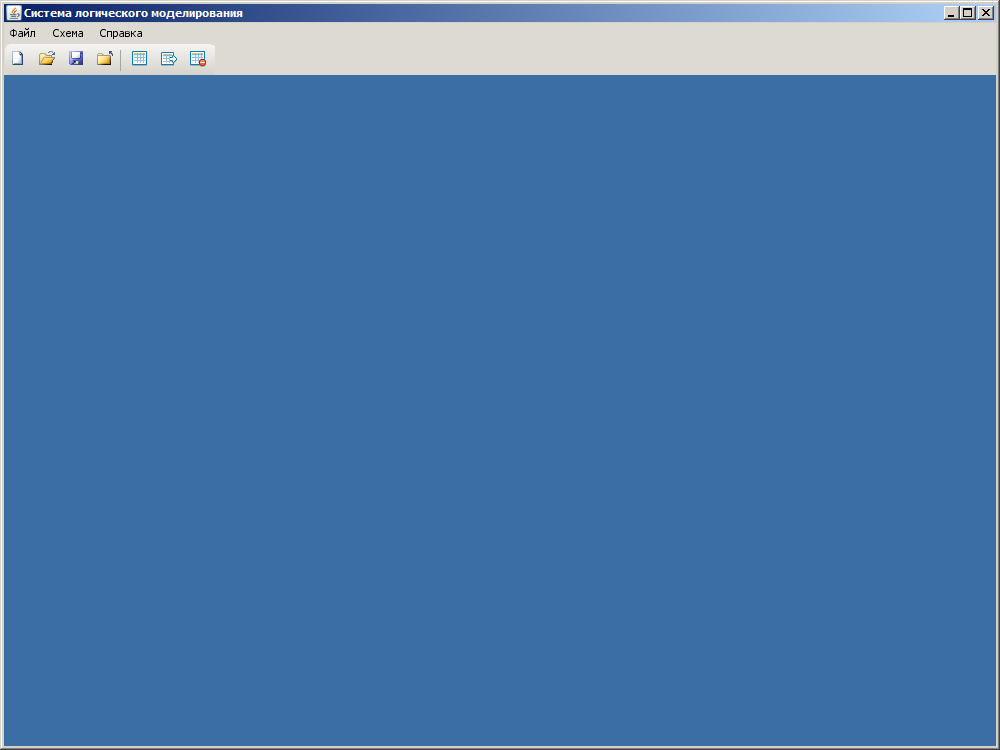


Рис. 3.1 – Главное окно программы

Главное окно содержит главное меню и главную панель инструментов. Программа дает возможность работать с множеством схем одновременно, так как поддерживает интерфейс MDI, и каждая схема имеет свой фрейм внутри главного окна.

Для создания новой схемы можно воспользоваться пунктом главного меню «Файл»-> «Новый…», комбинацией клавиш Ctrl-N или кнопкой на панели инструментов. После этого на экране появится диалоговое окно для указания настроек отображения схемы, которое изображено на рисунке 3.2.

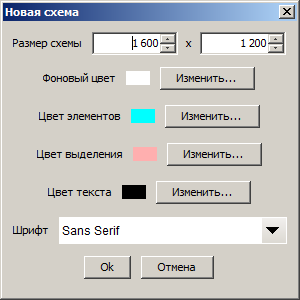


Рис. 3.2 – Диалоговое окно для выбора настроек отображения схемы

После этого в рабочей области главного окна программы откроется фрейм для работы со схемой, который изображен на рисунке 3.3.

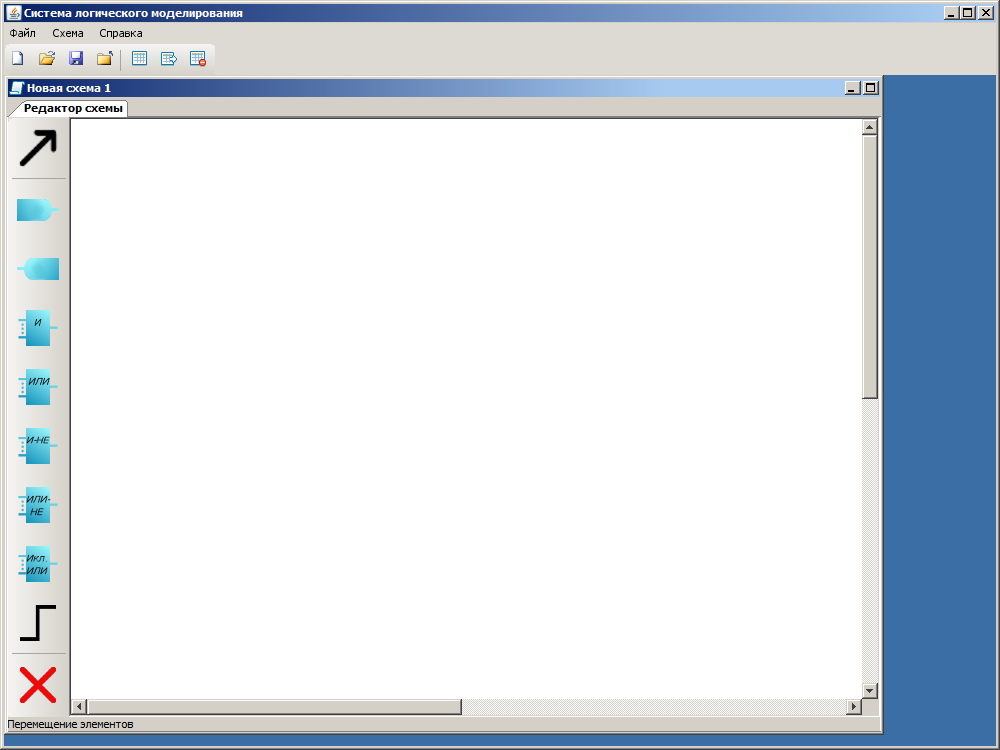


Рис. 3.3 – Главное окно программы с фреймом для работы со схемой

Слева во внутреннем фрейме находится панель инструментов для ввода схемы. С помощью кнопок этой панели можно выбрать необходимое действие. Непосредственно сам ввод производится с помощью мыши в рабочей области редактора схем с поддержкой механизма drag’n’drop. Для добавления элемента на схему необходимо выбрать необходимый элемент на панели инструментов и кликнуть на рабочей области. После этого появится диалоговое окно для ввода параметров элемента (рис. 3.4).

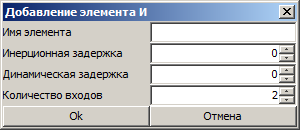


Рис. 3.4 – Диалоговое окно для ввода параметров элемента

Параметры элемента можно изменить позже, сделав на нем двойной клик. При добавлении элемента в недопустимое место будет выведено окно, сообщающее об ошибке (Рис. 3.5).

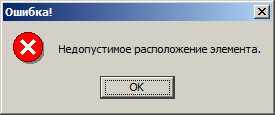


Рис. 3.5 – Диалоговое окно, сообщающее о ошибке ввода

Для соединения элементов необходимо выбрать данное действие на панели инструментов, нажать клавишу мыши на входе/выходе элемента и тянуть мышь к выходу/входу необходимого элемента. Возможные соединения будут подсвечиваться.

Для удаления элемента необходимо выбрать данное действие на панели инструментов и кликнуть на элементе и подтвердить данное действие в диалоговом окне.

Для сохранения схемы можно воспользоваться пунктом главного меню «Файл»-> «Сохранить», комбинацией клавиш Ctrl-S или кнопкой на главной панели инструментов. Схема сохраняется в XML-файл. Для открытия ранее созданной схемы необходимо воспользоваться пунктом главного меню «Файл»-> «Открыть…», комбинацией клавиш Ctrl-O или кнопкой на главной панели инструментов. Закрыть схему можно с помощью пункта главного меню «Файл»-> «Закрыть», комбинации клавиш Ctrl-W или кнопки на главной панели инструментов.

Пример созданной схемы приведен на рисунке 3.6.

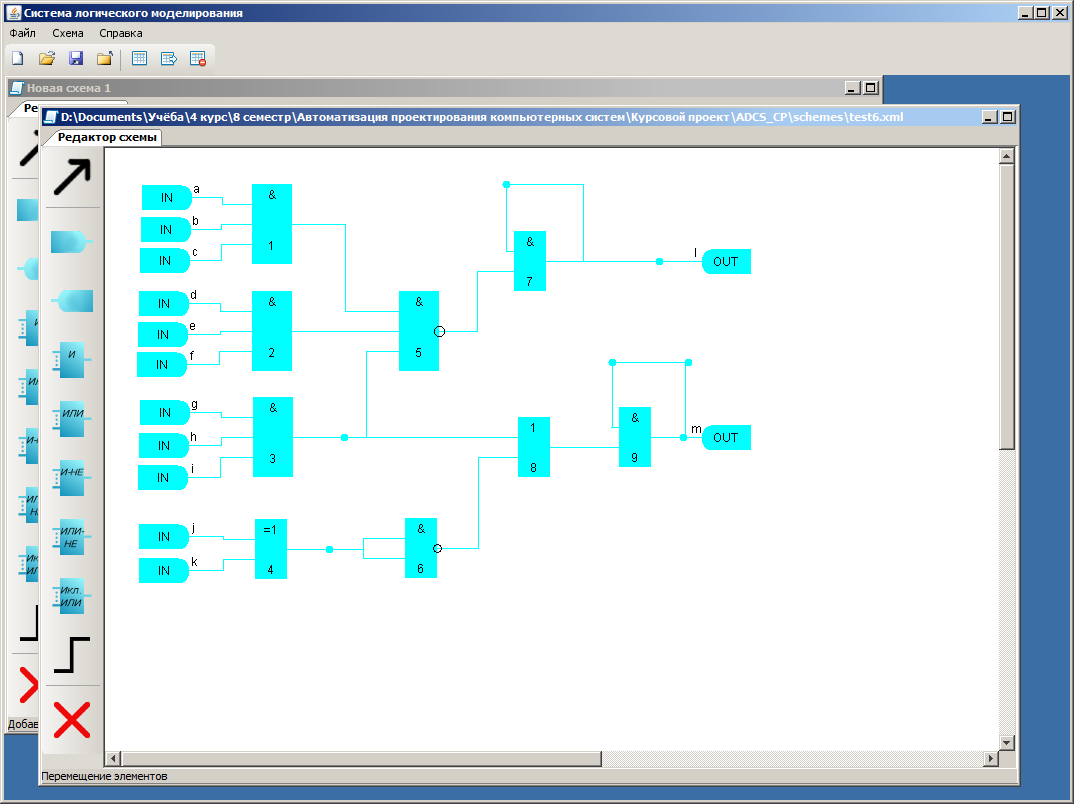


Рис. 3.6 – Пример созданной схемы

Для просмотра матрицы связей между элементами схемы необходимо воспользоваться пунктом главного меню «Схема»-> «Матрица связности», комбинацией клавиш Ctrl-Shift-M или кнопкой на главной панели инструментов. При этом матрица будет выведена в новой вкладке в фрейме схемы. Матрица связей между элементами схемы для схемы на рис. 3.6 приведена на рисунке 3.7.

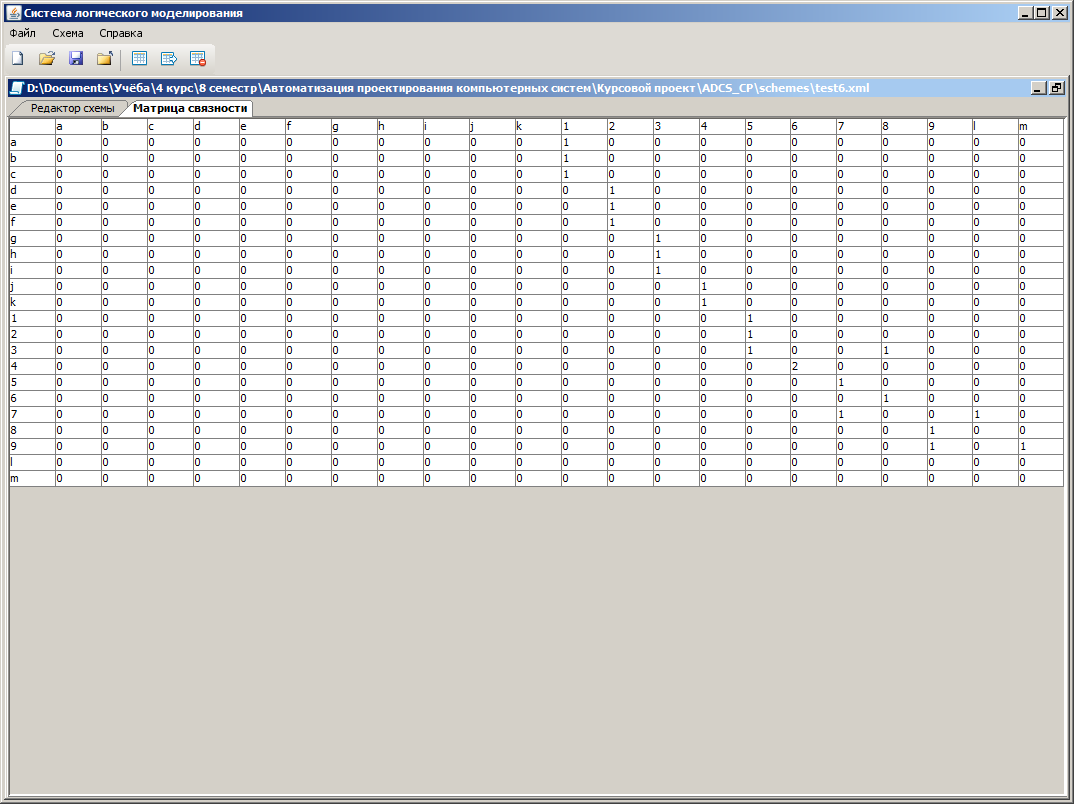


Рис. 3.7 – Матрица связей между элементами схемы с рис.3.6

Для перехода в режим моделирования нужно использовать пункт главного меню «Схема»-> «Моделирование», комбинацию клавиш Ctrl-M или кнопку на главной панели инструментов. При этом изменение схемы становится невозможным. Чтобы отредактировать схему нужно выйти из режима моделирования. Сделать это можно с помощью пункта главного меню «Схема»-> «Редактировать схему», комбинации клавиш Ctrl-E или кнопки на главной панели инструментов.

При переходе в режим моделирования для управления моделированием и вывода результатов откроется новая вкладка в фрейме схемы (рис. 3.8). Вверху находится панель управления моделирования. Ниже находятся таблица и временная диаграмма для вывода результатов.

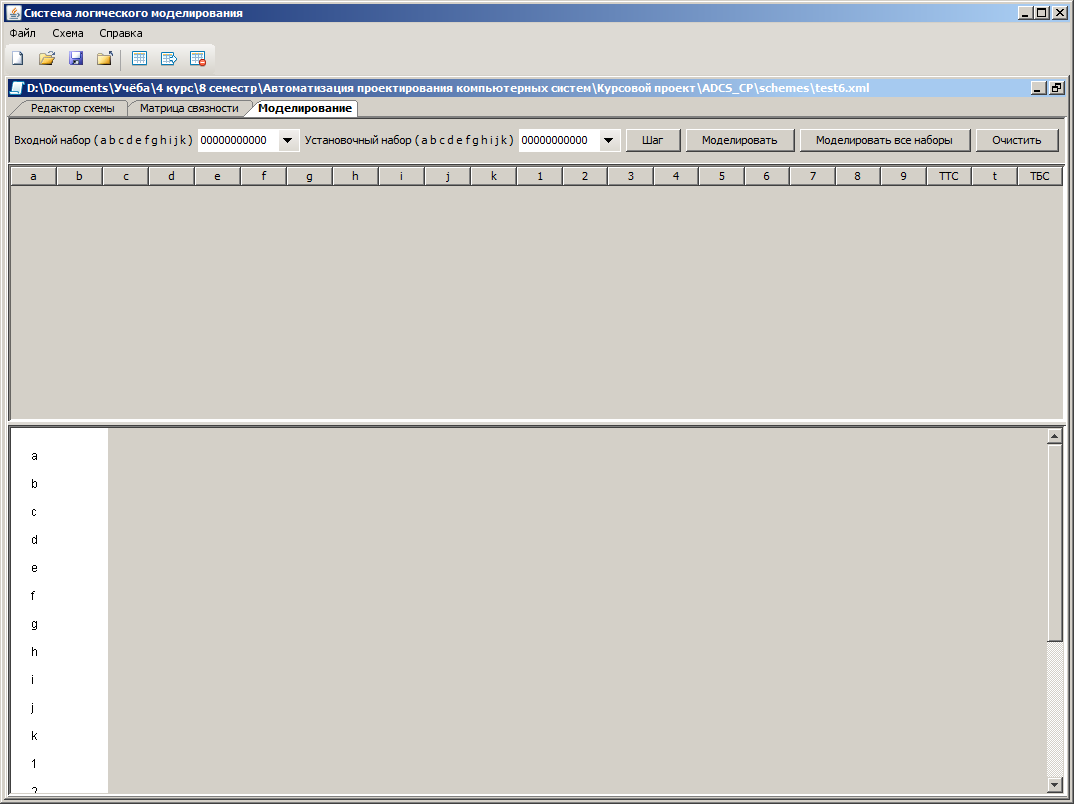


Рис. 3.8 – Вкладка режима моделирования схемы

Для начала моделирования необходимо выбрать из списка установочный входной набор, необходимый входной набор и нажать кнопку «Моделировать». Также имеется возможность моделировать схему пошагово с помощью кнопки «Шаг» и промоделировать схему на всех возможный входных наборах с помощью кнопки «Моделировать все наборы». Для завершения моделирования и очистки результатов нужно воспользоваться кнопкой «Очистить».

Примеры моделирования схемы с рис. 3.6 приведены на рисунках 3.9 и 3.10.

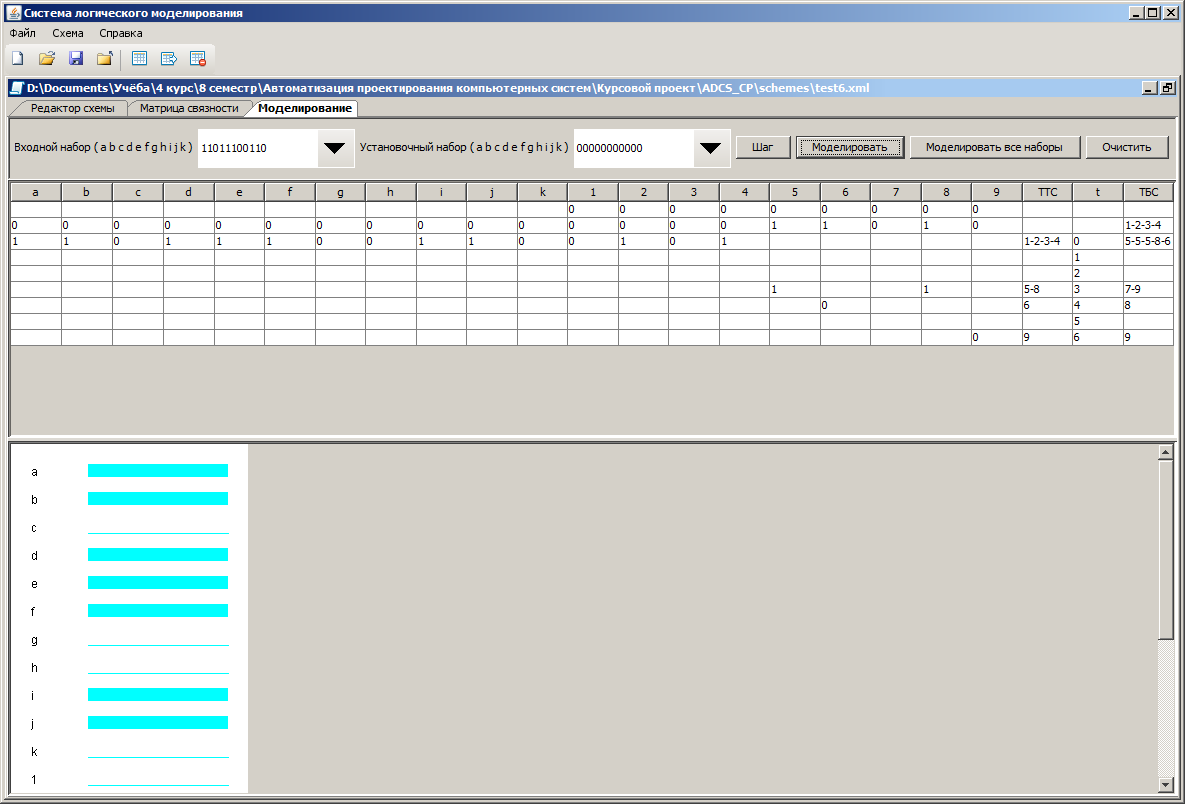


Рис. 3.9 – Пример моделирования схемы с рис.3.6

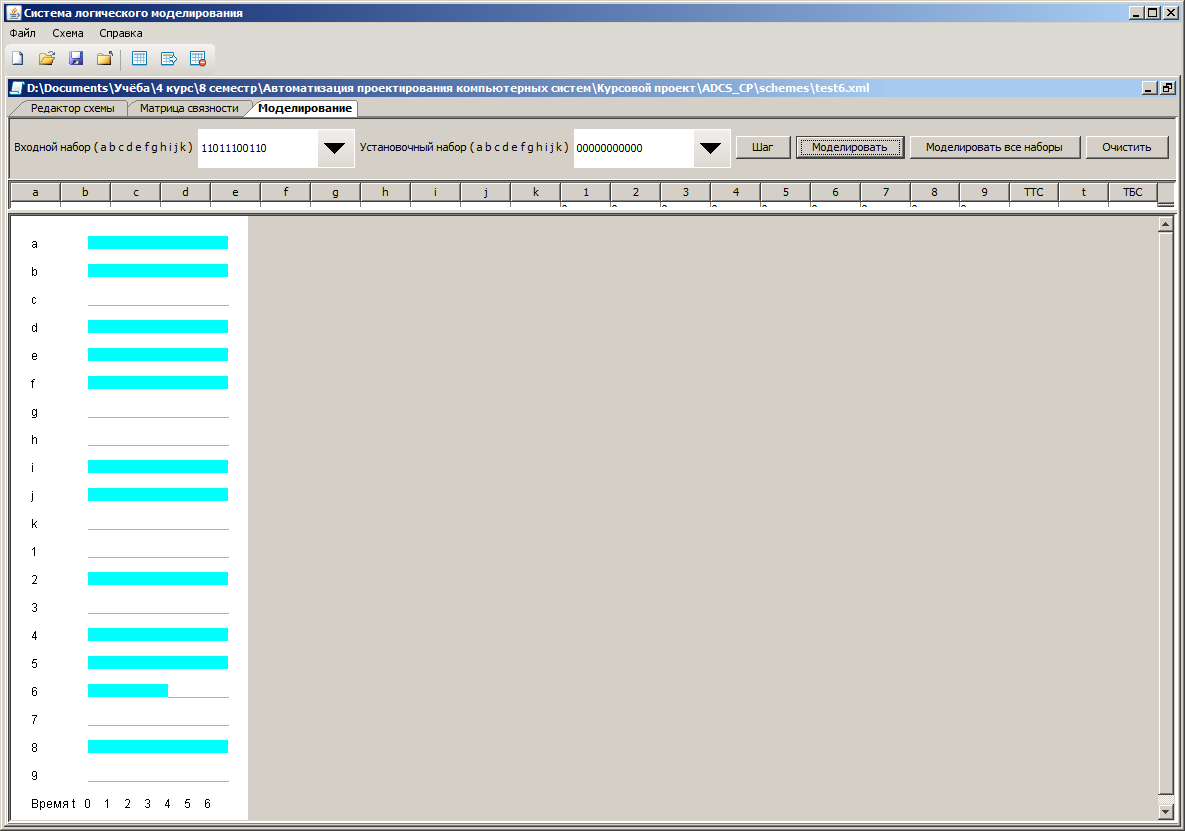


Рис. 3.10 – Временная диаграмма моделирования схемы с рис.3.6

Закрыть программу можно с помощью пункта главного меню «Файл»-> «Выход» или комбинации клавиш Ctrl-Q.

# *Заключение*

В данной работе была разработана программа, позволяющая производить моделирования логических схем с помощью событийного алгоритма (ЛИД-модель элементов).

Преимуществом использованного для моделирования алгоритма является то, что на каждом шаге производится моделирование не всех элементов схемы, а только тех, на входах которых возможно изменение сигнала. Следовательно, при моделировании с помощью данного алгоритма производится меньшее количество вычислений, чем при использовании итерационных алгоритмов. Также достоинством событийного алгоритма является то, что он довольно точно отображает поведение схемы, так как дает возможность учитывать динамические и инерционные задержки на элементах схемы.

Недостаток событийного алгоритма – это сложность реализации из-за интенсивного использования динамических таблиц текущих и будущих событий.

# *Список литературы*

1. Конспект лекций по курсу «Автоматизация проектирования компьютерных систем».
2. Норенков И.П. Основы автоматизированного проектирования // М.: Издательство МГТУ имени Н.Э. Баумана, 2002. – 336 с.
3. Жабін В.І., Жуков І.А., Клименко І.А., Ткаченко В.В. Прикладна теорія цифрових автоматів: Навч. посібник // К.: Книжкове видавництво НАУ, 2007. – 364 с.