Министерство образования Российской Федерации
Ульяновский государственный технический университет

На правах рукописи

Волков Сергей Вячеславович

УДК 681.32

ИССЛЕДОВАНИЕ И РАЗРАБОТКА СРЕДСТВ ПОДДЕРЖКИ ИЗУЧЕНИЯ СХЕМОТЕХНИЧЕСКОГО ПРОЕКТИРОВАНИЯ ЦИФРОВЫХ УСТРОЙСТВ

552808 – Информационное и программное обеспечение автоматизированных систем

Магистерская диссертация

Научный руководитель -
к.т.н., профессор Негода В.Н.

Ульяновск - 2001

Содержание

Введение 4

Глава 1. Обзор методов построения систем поддержки обучения, Интернет/Интранет технологий и существующих систем для изучения цифровой схемотехники 8

1.1. Компьютерные системы поддержки обучения как виртуальные образовательные структуры 8

1.1.1. Классификация компьютерных систем поддержки обучения . 8

1.1.2. Концепция интеллектуальной гипермедийной системы поддержки обучения 11

1.1.3. Интеграция интеллектуальных гипермедийных систем поддержки обучения в образовательные структуры 13

1.2. Обзор средств для разработки в WEB 18

1.2.1. Java-технология 18

1.2.2. ActiveX-технология 24

1.2.3. Сравнительный анализ технологий 28

1.3. Существующие системы поддержки изучения цифровой схемотехники 34

1.3.1. Основные проблемы, возникающие при изучении цифровой схемотехники 34

1.3.2. Учебная система «Последовательностные цифровые устройства» 35

1.3.3. Учебная система «Проектирование цифровых устройств на вентильных матрицах» 38

1.3.4. Проектирование цифровых устройств в системе HLCCAD .. 39

1.3.5. Micro-CAP и Micro-Logic - интегрированные системы моделирования логических и цифровых схем 42

1.3.6. САПР P-CAD: назначение и характеристика основных программ 44

1.3.7. Сравнительный анализ рассмотренных систем 46

1.4. Выводы по главе 47

Глава 2. Исследование подходов к построению учебных схемотехнических САПР и разработка структуры основных компонетов подсистемы поддержки изучения схемотехнического проектирования в ТССМ 49

2.1. Определение общего подхода к построению подсистемы поддержки изучения схемотехнического проектирования 49

2.2. Общая функциональная структура ТССМ и подсистемы поддержки изучения схемотехнического проектирования 52

2.3. Разработка модуля минимизации ФАЛ 55

2.3.1. Способы задания ФАЛ 55
2.3.2. Формы представления ФАЛ..... 56
2.3.3. Основные законы и тождества алгебры логики 57
2.3.4. Минимизация ФАЛ методом Квайна-Мак-Классски 57
2.3.5. Функциональная структура модуля минимизации ФАЛ 59
2.4. Разработка модуля тестирования цифровых схем на соответствие заданию 61
2.4.1. Основные подходы к тестированию цифровых схем 61
2.4.2. Метод тестирования цифровых схем в ТССМ и структура тестовых заданий 64
2.4.3. Функциональная структура модуля тестирования 66
2.5. Выводы по главе 68
Глава 3. Программные реализации модулей и экспериментальные исследования применимости средств поддержки изучения схемотехнического проектирования в ТССМ 69
3.1. Программная реализация модуля минимизации ФАЛ . 69
3.2. Программная реализация модуля тестирования цифровых схем 74
3.3. Разработка учебных цифровых схем и контрольных тестовых заданий 77
3.4. Требования обеспечения установки и эксплуатации 78
3.5. Измерение и сравнение производительности Java и C++ программ 79
3.6. Выводы по главе 84
Заключение 85
Литература 86
Приложение 1. Набор учебных цифровых схем 90
Приложение 2. Набор контрольных тестовых заданий для проверки работоспособности учебных цифровых схем 97
Приложение 3. Набор тестовых программ для оценки производительности 105

Введение

В настоящее время на мировом рынке существует большое количество оргтехники и промышленного оборудования, в основу которого положены принципы взаимодействия мелких цифровых узлов. При разработке таких узлов всегда необходимо знание базовых принципов цифровой схемотехники. В связи с этим учебные планы многих специальностей технических вузов содержат в себе дисциплины по схемотехническому проектированию. При изучении основ схемотехнического проектирования простейших цифровых устройств основным видом практической работы студентов является построение моделей таких устройств и исследование их поведения. В процессе построения моделей могут быть использованы либо учебные схемотехнические стенды (которые и использовались до недавнего времени в большинстве вузов) с последующим контролем знаний обучаемых преподавателем, либо учебные системы автоматизированного проектирования (САПР), которые позволяют строить программно-логические модели цифровых устройств, моделировать их функционирование и позволяющие организовать автоматический контроль знаний. Следует отметить, что в настоящее время в большинстве случаев выбор останавливается на последних, и на то есть веские причины. Совершенно ясно, что учебные САПР обладают гораздо большей функциональностью, а это очень важно в процессе обучения. С помощью САПР, к примеру, мы можем смоделировать работу практически любого устройства, причем не только уже существующего, но и разрабатываемого. САПР, как правило, обладает более удобным интерфейсом взаимодействия с пользователем и может гибко настраиваться в зависимости от круга решаемых задач. Кроме того, использование САПР в учебных целях экономически выгоднее, поскольку на сегодняшний день вузы уже оснащены высокопроизводительной компьютерной техникой в достаточном объеме.

Несмотря на то, что уже имеется довольно много профессиональных схемотехнических САПР, для вузов задача создания собственных САПР пока остается актуальной. Большей частью это связано с тем, что все

доступные разработки, как правило, ориентированы для использования на производстве специалистами в данной области и мало пригодны для применения в учебном процессе. Интеграция учебных курсов и подсистем контроля знаний обучаемых в промышленные САПР также крайне затруднительна. В этой связи интерес представляет не только задача создания конкретной учебной САПР, но и задача исследования общих вопросов построения программно-логических моделей учебных САПР, поскольку проблемы выбора структуры таких программ практически не освещены в научной и технической литературе.

При рассмотрении данной задачи в рамках проблематики создания средств поддержки учебного процесса целесообразно ориентироваться на развивающуюся систему. Появление новых технологий в области программирования дает возможность разработки систем функционирующих в телекоммуникационной среде. Такая ориентация дает не только возможность централизованного хранения и обновления версий продукта, но и предполагает активный процесс развития и модификации продукта в ходе его эксплуатации с привлечением к разработке и тестированию широкого круга программистов и пользователей.

Цель работы заключается в исследовании возможных методов для создания систем поддержки обучения и учебных телекоммуникационных САПР, анализе существующих решений подобного рода и разработке программно-логических моделей средств поддержки изучения схемотехнического проектирования цифровых устройств в телекоммуникационной системе схемотехнического моделирования (ТССМ).

В соответствии с поставленной целью в работе формулируются и решаются следующие задачи:

1. Анализ методологии построения систем поддержки обучения и учебных САПР;
2. Рассмотрение общих вопросов реализации таких систем;
3. Рассмотрение вопросов реализации систем поддержки обучения и учебных САПР в телекоммуникационной среде;
4. Сравнительный анализ функциональных возможностей существующих учебных систем для

изучения цифровой схемотехники и формирование требований к функциональным характеристикам средств поддержки обучения в разрабатываемой САПР;
5. Формирование подходов к взаимодействию с обучаемыми;
6. Определение структуры программной модели средств поддержки обучения;
7. Разработка каркаса (иерархии классов) программно-логической модели;
8. Экспериментальная разработка программно-логической модели подсистемы поддержки изучения схемотехнического проектирования в ТССМ с целью проверить сформированный подход и исследовать следующие параметры подсистемы:
   - адекватность полученной программной модели;
   - функциональность и «дружественность» интерфейса с пользователем по сравнению с существующими решениями;
   - быстродействие, устойчивость, плюсы и минусы использования телекоммуникационной среды.

Предметом исследования являются методы построения учебных систем и разрабатываемая программно-логическая модель подсистемы для поддержки изучения схемотехнического проектирования цифровых устройств в ТССМ.

Метод исследования базируется на анализе методологии построения систем поддержки обучения и учебных САПР, технологии проектирования сетевых программных систем, а также на обработке и анализе результатов экспериментов.

Научная новизна работы состоит в формулировке и исследовании подхода к проектированию программно-логических моделей систем поддержки обучения и учебных САПР, ориентированных на использование в процессе изучения основ схемотехнического проектирования в телекоммуникационной среде.

Практическая ценность работы заключается в следующем:

- сформулирован подход к созданию средств поддержки изучения схемотехнического проектирования в телекоммуникационной среде;
- разработан каркас (иерархия базовых классов) программно-логической модели подсистемы поддержки изучения схемотехнического проектирования;
- реализована подсистема поддержки изучения основ схемотехнического проектирования, пригодная для использования в процессе обучения;
- подсистема снабжена дружественным интерфейсом и интегрирована в ТССМ.

Глава 1. Обзор методов построения систем поддержки обучения, Интернет/Интранет технологий и существующих систем для изучения цифровой схемотехники

1.1. Компьютерные системы поддержки обучения как виртуальные образовательные структуры

1.1.1. Классификация компьютерных систем поддержки обучения

Термин «компьютерное обучение» является собирательным понятием, в основе которого лежат два классификационных признака: стратегия обучения и тип обучающей программы [1]. Чтобы показать взаимосвязь этих признаков необходима классификационная схема существующих компьютерных систем поддержки обучения. Известные классификационыые схемы [2] являются либо крайне упрощенными, либо наоборот, очень сложны и многомерны. На мой взгляд наиболее предпочтительной является двумерная классификационная схема, представленная на рис. 1.

Рис. 1. Классификация компьютерных обучающих систем

Таким образом, классификация компьютерных обучающих программ зависит от двух признаков: стратегии управления обучением и поведения обучаемого. Альтернативными стратегиями обучения являются жесткозапрограммированный алгоритм (по существу без учета индивидуальных особенностей обучаемого), адаптивная программа, достаточно гибко реагирующая и контролирующая действия обучаемого, и,

наконец, программа, управляемая самим обучаемым. С другой стороны, альтернативными линиями поведения обучаемого являются:

- Пассивное поведение, когда обучаемый отвечает на вопросы;
- Рецептивное поведение, когда обучаемый активно усваивает информацию, но делает это не творчески;
- Поведение, основанное на сотворчестве, базирующееся на когнитивной теории обучения, когда обучаемый активно осмысливает свои знания и синтезирует новые знания;
- Поведение, основанное на готовности к новым действиям.

Итак, на классификационной схеме представлены основные 7 типов компьютерных обучающих систем. Оставляя за рамками данной работы детальную характеристику всех указанных типов систем, отметим, что на практике можно комбинировать различные классы систем, например, обучение вести с помощью гипермедийной системы, а для контроля знаний использовать экзаменационную систему задач или упражнений (репетиторство) и т.п. [3].

Остановимся несколько подробнее на анализе интеллектуальных и гипермедийных систем поддержки обучения, поскольку они, на мой взгляд, должны составлять ядро концепции виртуального образовательного пространства [4, 5].

Особенность интеллектуальных систем поддержки обучения состоит в том, что они должны быть способны, определив индивидуальные особенности обучаемого, самостоятельно спланировать стратегию обучения, контроля и подкрепления знаний [5]. Основой для проектирования подобного рода систем являются методы искусственного ителлекта. Архитектура и принципы построения интеллектуальных систем поддержки обучения могут быть различными, однако все они, как правило, включают в себя базу знаний (БЗ) предметной области, БЗ обучаемого, экспертную систему (ЭС) планирования процесса обучения и модуль интерфейса [5].

БЗ обучаемого играет ключевую роль в интеллектуальной системе поддерки обучения. Она должна быть динамической и модифицируемой. В

зависимости от ее состояния в дальнейшем планируется стратегия обучения. Принято различать следующие модели БЗ обучаемого [7]:

- **Стереотипная модель.** Стереотип включает несколько типов обучаемых и, если у обучаемого обнаруживается несколько характеристических признаков, то его относят к определенному типу (например, при обучении программированию различают начинающих программистов, продвинутых программистов и профессионалов);
- **Думающая модель.** Знания обучаемого воспринимаются как некоторое подмножество знаний из БЗ предметной области, которая в свою очередь, состоит из отдельных относительно небольших порций знаний. Тогда стратегия обучения состоит в постепенном наращивании БЗ обучаемого. Неверно понимаемая информация в этой модели не учитывается;
- **Модель ошибок.** Эта модель дополняет предыдущую тем, что дополнительно к экспертным знаниям добавляется так называемая библиотека ошибок, содержащая типичные ошибки и недоразумения, что позволяет более тонко планировать обучение;
- **Имитационная модель.** Эта модель предусматривает исследование когнитивного состояния обучаемого с целью анализа его знаний и будущего поведения на основе изучения его реакций и ошибок. Модель должна уметь сымитировать поведение обучаемого, например, при решении неизвестной ему проблемы.

Особенностью класса гипермедийных систем поддержки обучения является хранение знаний небольшими порциями информации (узловые точки), которые связаны друг с другом посредством гиперссылок. Это позволяет генерировать любые, в особенности нелинейные информационные структуры [8]. В качестве узлов могут, например, фигурировать текст, рисунок, график и т.п. Гипермедийная программа придает этой информации динамизм с помощью звука, анимации и видеофрагментов. Последовательностью обучения может управлять сам обучаемый с помощью средств навигации.

На основе некоторого факта обучаемый может проследить контекст и ссылки на другие факты. Связанные с данным фактом понятия могут быть проиллюстрированы графически и «оживлены» путем анимации. Гипермедийные обучающие системы основаны на идее создания новых форм представления знаний, которые позволяют существенно упростить обмен знаниями между системой поддержки обучения и человеком.

1.1.2. Концепция интеллектуальной гипермедийной системы поддержки обучения

Предполагается, что реализация проектов гипермедийных систем поддержки обучения позволяет повысить эффективность дистанционного обучения и самообразования. Не буду повторять известные по многочисленной литературе преимущества и недостатки методик дистанционного образования. Хочется указать лишь следующие особенности, связанные с применением интеллектуальных гипермедийных систем поддержки обучения в сфере дистанционного образования:

- Обучаемый самостоятельно определяет темп обучения, тип программы и предметную область;
- Длительность «урока» зависит от состояния обучаемого в данный момент;
- Беспроблемная возможность повторения того или иного материала;
- Глубокий отпечаток знаний благодаря интерактивности и когнитивной графике;
- Высокая мотивация благодаря адаптивному поступлению порций информации;
- Стабильность нагрузки на обучаемого, поскольку качество процесса обучения не зависит от кондиции учителя.

Известный классический подход к построению гипермедийных систем [8] базируется на том, что вся структурная и содержательная информация импортируется в среду извне, а затем данные конвертируются в специальные форматы. Доступ к содержащейся внутри системы информации через внешние приложения становится невозможным из-за того, что файлы имеют свои внутренние форматы. Подобного рода подход имеет еще один недостаток – среда становится зависимой от

используемой программной платформы. Следовательно, импортировать результаты на другие платформы либо невозможно, либо потребует значительных затрат. Другой недостаток классического подхода заключается в некоторой избыточности администрирования информации. Речь идет о том, что идентичное содержание, многократно используемое в среде, избыточно запоминается не как ссылка на информацию, а как отдельный объект. Это означает, что при изменении содержания каждый из найденных системой объектов должен быть актуализирован.

Между тем, отличительной чертой интеллектуальных гипермедийных программ является большой объем демонстрационной графики. Для упрощения процесса портирования и модификации среды обучения все графические и текстовые данные должны запоминаться и конвертироваться в стандартные форматы. Демонстрация графики и доступ к текстам можно производить с помощью относительно небольших программ пользовательского интерфейса (Front-End-Program). Такой подход выгодно отличается от классического подхода описанного выше тем, что требует значительно меньших затрат на экспорт презентационных модулей в другие платформы и системы. Дополнительно к «использованию» среды, при котором на некотором компьютере хранится как текстовая, так и графическая часть программы обучения, появляется возможность преобразовывать учебную информацию централизованно на сервере с помощью программ пользовательского интерфейса вызываемых, например, через Internet. Другое преимущество предлагаемой концепции заключается в гибком использовании информации из БЗ предметной области для разных целей, например, для самообучения, для лекционного курса, для справочных целей и т.п. Таким образом, структура среды поддержки обучения является трехуровневой: программы пользовательского интерфейса, например, Java-программы; реляционный банк данных; медийные файлы в стандартном формате. Программы пользовательского интерфейса служат для презентации содержания предметной области и управления диалогом с обучаемым. Язык Java позволяет экспортировать презентационные компоненты без дополнительных усилий в различные платформы.

Далее информация о структуре, требованиях и связях с компонентами системы поддержки обучения запоминается в реляционном банке данных. Доступ презентационной программы к банку данных может быть осуществлен, например, с помощью независимого интерфейса ODBC (Open DataBase Connectivity) или JDBC (Java DataBase Connectivity) [15]. Задача интерфейса заключается в том, чтобы на базе языка SQL осуществить DBMS-независимый обмен данными между интерфейсным приложением и банком данных. С помощью ODBC или JDBC интерфейса можно связать интерфейсные приложения с такими базами данных, как ORACLE, Sybase, Microsoft, IBM, MySQL [12] и т.д.

Отдельные медийные файлы, из которых состоит система поддержки обучения, хранятся не только в уже упомянутом банке данных, но и являются независимыми друг от друга файлами. В банке данных заполняются ссылки на текущие медийные файлы. Другая причина для внешней конвертации файлов данных состоит в простоте способа их изменения с помощью известных программ обработки графики и, звука и текста.

1.1.3. Интеграция интеллектуальных гипермедийных систем поддержки обучения в образовательные структуры

Основой виртуальных образовательных структур, на мой взгляд, должны быть интегрированные интеллектуальные гипермедийные среды поддержки обучения и телекоммуникационные технологии. Все учебно-методические материалы (тренажерные системы, видеоматериалы, анимационная и имитационная графика, тексты лекционных курсов) представляются в виде гипермедийных документов. Выдача материала производится по сети, e-mail или обычной почте. В библиотеке интегрированы электронные версии книг и статей с доступом к ним в режиме on-line. Связь через телеконференции поможет решить проблемы групповой работы обучаемых.

Все это открывает новые перспективы для студентов виртуального образовательного пространства.

Прежде чем детализировать структуру такого виртуального образовательного пространства, поясним возможные сценарии и роли его участников. Возможны следующие варианты статуса пользователя: гость;

обучающийся; преподаватель и автор курса; исследователь.

Права гостя ограничены справочной информацией и демонстрационными версиями учебных курсов. Обучающийся имеет свой электронный адрес, доступ к учебным программам на текущий семестр, возможность получить разнообразные консультационные и библиотечные услуги. Преподаватель или автор курса осуществляет роль консультанта и координатора индивидуальной самостоятельной работы студента. Исследователь в виртуальной образовательной системе, например, дипломник имеет возможности широкого поиска интересующей его информации, в том числе через www-сервер.

В целом, архитектура виртуального университета базируется на двух платформах: технической и пользовательской. Техническая платформа включает сетевую структуру и системное программное обеспечение. Пользовательскую платформу образуют разнообразные учебно-методические программы, учебная документация и т.п., которая в отличии от технической платформы зависит от целей пользователя и может влиять на состав технической платформы.

Техническая платформа должна базироваться на системном программном обеспечении, включающем разнообразные Internet-услуги [9]. Перечислим их:

- WWW-сервер является основой гипермедийного независимого пользовательского интерфейса в виртуальном университете;
- FTP-сервер является альтернативой www при передаче, например, демонстрационных программ или больших пакетов данных;
- Mail-сервер управляет передачей сообщений между пользователями сети виртуального образовательного пространства и хранит сообщения электронной почты. Этот сервер разрешает асинхронный обмен любыми документами;
- News-сервер служит для организации дискуссий и рабочих консультаций;

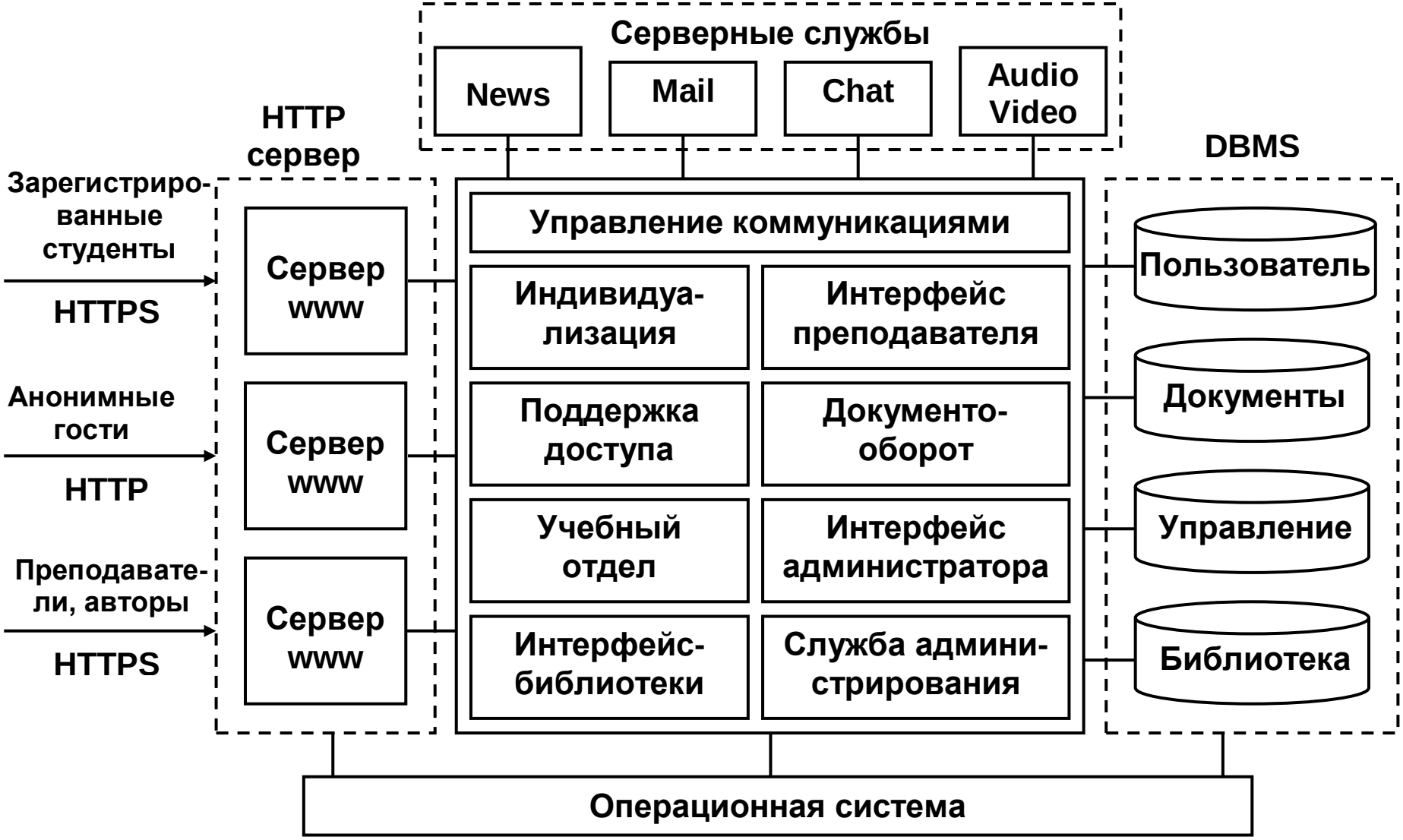


Рис. 2. Общая архитектура системы поддержки обучения

- Chat-сервер обеспечивает обмен информацией (текстовый диалог) в реальном времени, разговор в сети Internet сразу нескольких участников, причем текст, введенный каждым из них, отображается одновременно на экране у всех участников;
- Сервер видеоконференций обеспечивает магистраль многоадресной передачи аудио- и видеоинформации из одной точки нескольким выделенным компьютерам сети в отличие от широковещательного режима, когда пересылка информации идет всем узлам сразу.

Тот факт, что для организации технической платформы виртуального образовательного пространства могут применяться уже существующие информационные системы и технологии, является преимуществом и, одновременно, недостатком при реализации технической платформы университета. Система, управляющая виртуальным университетом, должна уметь устанавливать контекстно зависимые каналы связи к каждой изучаемой дисциплине. Общая архитектура системы поддержки обучения может иметь вид, представленный на рис. 2. Гибкость системы обеспечивается, например, применением CGI-интерфейса.

Опишем более подробно отдельные компоненты представленной общей архитектуры системы поддержки обучения. Особенности взаимодействия системы управления базами данных (DBMS) с www-сервером общеизвестны. Документы на языке гипертекстовой

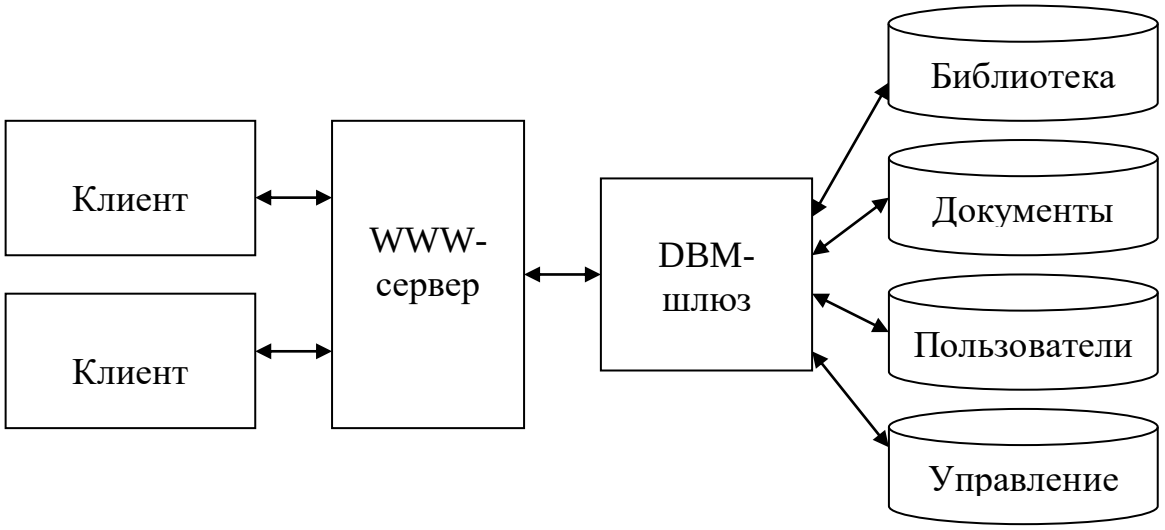


Рис. 3. Трехуровневая система взаимодействия «клиент-банк данных»

разметки (HTML) внутри www- браузера не обрабатываются,они служат лишь для ввода и вывода информации. Эта особенность HTML приводит к трехуровневой архитектуре взаимодействия пользователя с банком данных (рис. 3).

Подобного рода схема взаимодействия, в отличии, от известной двухуровневой схемы [11] позволяет повысить информационную безопасность данных, снизить нагрузку сети, увеличить гибкость системы.

Известно, что протокол HTTP предусматривает передачу данных по сети «открытым» текстом. Это делает возможным манипулирование информацией между клиентом и сервером и снижает информационную безопасность в сети системы поддержки обучения. Для защиты информации можно применить www-сервер «безопасности», использующий разнообразные методы защиты с помощью ключей, что гарантирует невозможность манипулирования данными в ходе их передачи по сети [7].

Обсуждаемая система поддержки обучения предусматривает открытый интерфейс, который должен быть независим от возможных изменений в системе. Для этого имеются два альтернативных варианта. Первый вариант основан на механизме тиражирования и автоматизированном обмене данными между базами данных администратора и DBMS.

Второй вариант возможен, если применяется не трехуровневая схема взаимодействия «клиент - банк данных», а двухуровневая схема, в которой отсутствует DBMS-шлюз со всеми вытекающими отсюда негативными последствиями.

Сервер администратора решает проблему интеграции всех базовых компонент системы и располагается на www-сервере.

Сервер администратора требует наличия двух различных интерфейсов. Один из них отвечает за конфигурирование базовых компонент системы и базируется на интерфейсе прикладного программирования CGI, используемом в Internet. CGI определяет взаимодействие программы создания динамических www-документов и HTTP-сервера. Другой интерфейс выполняет собственно задачи администрирования банков данных пользователей и документов. Административный сервер является основой интерфейса преподавателя курса.

Возникает вопрос об организации клиент-серверных приложений для коммуникационного управления. Клиент реализует, в основном, интерфейс пользователя и протокол коммуникаций. Сервер, располагая таким же протоколом коммуникаций, реализует логику конкретного приложения. Выбор в качестве инструментального средства, например, языка Java дает следующие преимущества при реализации коммуникационного управления:

- Имеется специальный механизм, который позволяет выполнять Java-программу внутри www-браузера;
- Библиотечные классы Java позволяют соединять реляционные банки данных с Java-программой;
- Библиотечные классы Java поддерживают сетевые приложения.

1.2. Обзор средств для разработки в WEB

1.2.1. Java-технология

Язык Java™ - это относительно новый объектно-ориентированный язык программирования [14, 15, 16, 17, 18], созданный фирмой Sun для разработки программ, распространяемых по сети Internet. Система программирования Java позволяет использовать World Wide Web (WWW) для распространения небольших интерактивных прикладных программ (апплетов), которые размещаются на серверах Internet, транспортируются клиенту по сети (точно так же, как картинки или звуковые файлы), автоматически устанавливаются и запускаются на месте, как часть документа WWW. При этом апплет имеет весьма ограниченный доступ к ресурсам компьютера клиента, так что он может предоставить произвольный мультимедийный интерфейс и выполнять сложные вычисления, не привнося при этом риска заражения вирусом или порчи данных.

Система программирования Java может служить основой для совместной разработки больших программных систем коллективом разработчиков, связанных между собой только через WWW (они и знакомы между собой могут быть лишь заочно, через e-mail, а когда они наконец повстречаются где-нибудь на международном симпозиуме, в их активе уже может быть совместно

разработанная программная система). Java и WWW являются первыми системами, обеспечивающими такую возможность, поэтому их внедрение и распространение многие программисты справедливо называют революцией в разработке программного обеспечения. Ясно, что ведущую роль в обеспечении указанной возможности играет именно Java, так как именно Java позволяет распространять не просто тексты, а работающие программы и их фрагменты (апплеты) по WWW.

Java проста, объектно ориентированна и знакома. Система Java создана на основе «простого» языка программирования, техника использования которого близка к общепринятой и обучение которому не требует значительных усилий. Java как язык программирования является объектно ориентированной с момента основания. Кроме того программист с самого начала обеспечивается набором «стандартных» библиотек, обеспечивающих функциональность от стандартного ввода/вывода и сетевых протоколов до графических пользовательских интерфейсов. Эти библиотеки легко могут быть расширены. Несмотря на то, что язык C++ был отвергнут, синтаксис языка Java максимально приближен к синтаксису C++. Это делает язык знакомым широкому кругу программистов. В то же время из языка были удалены многие свойства, которые делают C++ излишне сложным для пользования, не являясь абсолютно необходимыми. В результате язык Java получился более простым и органичным, чем C++.

Надежность и безопасность. Java существенно облегчает создание надежного программного обеспечения. Кроме исчерпывающей проверки на этапе компиляции, системой предусматривается анализ на этапе выполнения. Сам язык спроектирован так, чтобы вырабатывать у программиста привычку писать "правильно". Модель работы с памятью, в которой исключено использование указателей, делает невозможными целый класс ошибок, характерных для C и C++. В силу того, что Java предназначена для работы в распределенной среде, безопасность становится чрезвычайно важной проблемой. Требования безопасности определяют многие черты как языка, так и реализации всей системы.

Независимость от архитектуры и переносимость. Компилятор Java производит байт-коды, т.е. модули

приложения имеют архитектурно-независимый формат, который может быть проинтерпретирован на множестве разнообразных платформ. Это уже не исходные тексты, но еще не платформно-зависимые машинные коды.

Следующий шаг - "замораживание" стандарта на формат основных встроенных типов данных. Программа, созданная на одной платформе, работает на всех остальных.

Этот стандарт фиксирован в документе, описывающем Java Virtual Machine. Стандарт может быть реализован на любой аппаратно-программной платформе, поддерживающей многопотоковость.

Производительность. Схема работы системы и набор байт-кодов виртуальной машины Java таковы, что позволяют достичь высокой производительности на этапе выполнения программы:

- анализ кодов на соблюдение правил безопасности производится один раз до запуска кодов на выполнение, в момент выполнения таких проверок уже не нужно, и коды выполняются максимально эффективно;
- работа с базовыми типами максимально эффективна, для операций с ними зарезервированы специальные байт-коды;
- методы в классах не обязательно связываются динамически;
- автоматический сборщик мусора работает отдельным фоновым потоком, не замедляя основную работу программы, но в то же время обеспечивая своевременный возврат свободной памяти в систему;
- стандарт предусматривает возможность написания критических по производительности участков программы в машинных кодах;

Интерпретируемый, многопотоковый и динамический. Интерпретируемая природа языка позволяет сделать фазу линкования простой, инкрементальной и, следовательно, быстрой. Это резко сокращает цикл разработки и тестирования программных фрагментов. Многопотоковость позволяет выполнять в рамках одного приложения несколько задач одновременно. Это становится особенно актуально в современных распределенных приложениях, когда процессы сетевого обмена могут идти одновременно и асинхронно. При этом программа

продолжает реагировать на ввод информации пользователем без неприятных задержек. Многопотоковость поддерживается на уровне языка - часть примитивов синхронизации встроена в систему реального времени, а библиотека содержит базовый класс Thread. К тому же системные библиотеки написаны thread-safe, т.е. все они могут быть использованы в многопотоковых приложениях. Система обеспечивает динамическую сборку программы. Классы подгружаются по мере необходимости, причем загружены они могут быть с любой точки сети, что позволяет сделать внесение изменений в приложения прозрачным для пользователя. Пользователь может быть уверен, что всегда работает со свежей версией приложения.

Обеспечение безопасности при распространении апплетов. Каждый апплет выбирается на компьютер его пользователя из Web с помощью браузера (авторы системы Java изготовили для этого свой браузер HotJava, однако большинство пользователей предпочитает Internet Explorer и Netscape). После этого он загружается, верифицируется и запускается на JavaVM на компьютере пользователя, имея доступ только к JavaVM и к памяти, которую он сам занимает (рис. 4). Ни к каким другим ресурсам компьютера

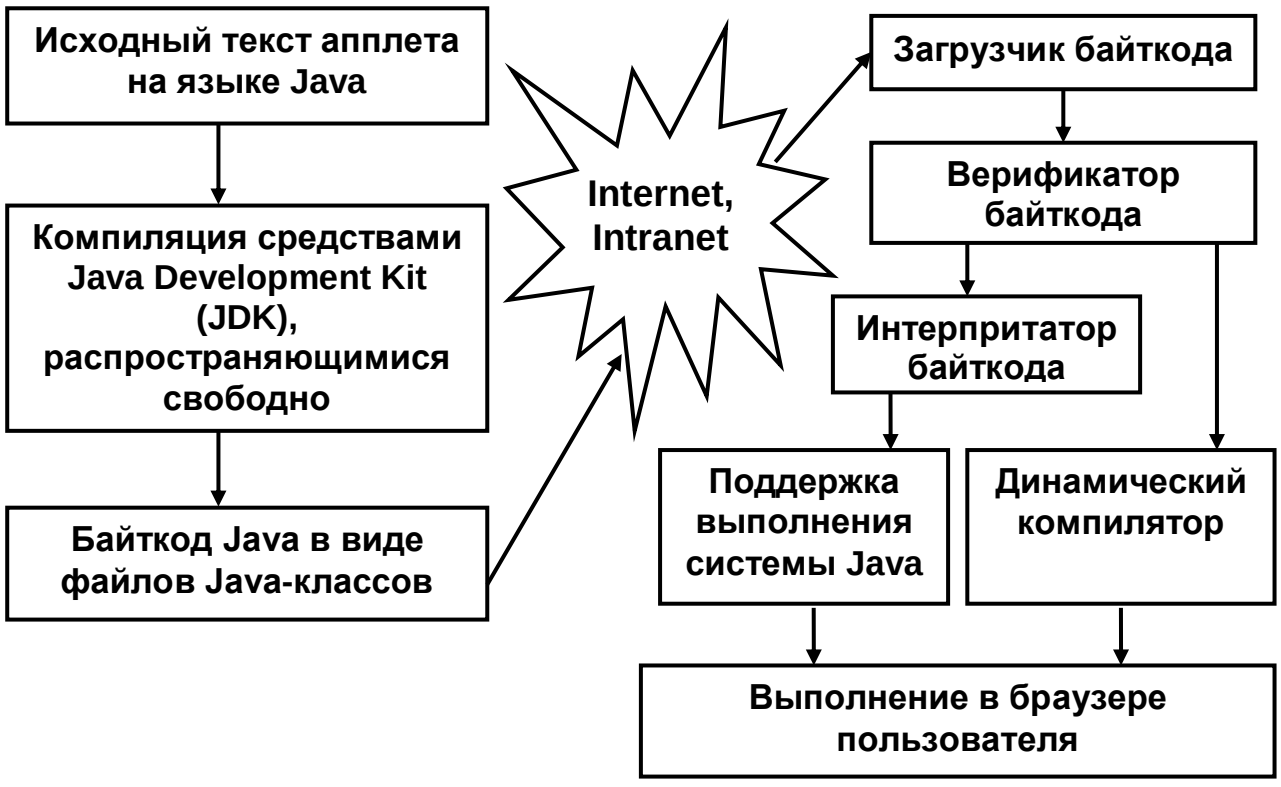


Рис. 4. Java-технология

пользователя апплет доступа не имеет, что обеспечивает защиту данных и других программ пользователя от порчи и предохраняет компьютер пользователя от заражения вирусами. Защита ресурсов компьютера, на котором выполняются апплеты, загружаемые из сети, является предметом особых забот разработчиков системы Java. Поэтому в системе поддерживается четыре уровня безопасности Java-программ (в частности, апплетов).

Первый уровень реализован в компиляторе на байткод. В отличие от традиционных компиляторов C/C++, в которых память распределяется во время компиляции, в компиляторе Java распределение памяти отложено до времени выполнения. Это, естественно, замедляет интерпретацию Java-программ, но это мешает потенциальным хаккерам взламывать систему, так как доступ к исходному коду Java-программы не дает возможности организовать несанкционированный доступ к ее памяти. Это связано и с тем, что в языке Java вместо указателей используются символические объектные ссылки в кучу, причем доступ к какому-либо объекту всегда ограничивается этим объектом.

Второй уровень защиты осуществляет верификатор байткода, который просматривает загруженный апплет перед его интерпретацией. ***Верификатор байткода*** – это несложная программа для доказательства теорем, которая проверяет загруженный байткод на удовлетворение им следующих условий (эти условия могут быть нарушены из-за неполадок в работе сети и при использовании разработчиком апплета недоброкачественного компилятора Java):

- байткод не выдает ссылок (указателей) на объекты, расположенные вне JavaVM;
- байткод не нарушает ограничений по доступу к ресурсам компьютера пользователя апплета;
- доступ к любым объектам апплета осуществляется в соответствии с их типом (строгая типизация);
- все параметры инструкций JavaVM имеют правильные типы;
- доступ к полям объектов апплета согласован с их определением как public, private, или protected.

Такая верификация байткода стала возможной благодаря выбранной семантической модели языка Java,

принятой схеме распределения памяти и сохранению информации о типах (классах) объектов Java-программы на период ее интерпретации.

Третий уровень защиты осуществляется при динамической загрузке классов при выполнении апплета на компьютере пользователя. Во время выполнения апплета он может вызывать различные классы из локальной файловой системы, а также классы с различных серверов сети. Для обеспечения большей безопасности каждый класс, загруженный из сети, выполняется в своем отдельном пространстве имен (пакете). Для классов, загруженных из локальной файловой системы, организуется общее пространство имен. Когда класс ссылается на какой-либо другой класс, загрузчик сначала пытается разрешить ссылку в пределах локального пространства имен, а затем в остальных (внешних) пространствах имен. Поэтому ни один из локальных классов не может "испортить" какой-либо внешний класс в сети. По этой же причине локальные классы не могут ошибочно ссылаться на классы из внешних (импортированных) пространств имен.

Наконец, **четвертый уровень** защиты связан с такой организацией интерфейсов с сетевыми протоколами (HTTP, FTP и др.), которые обеспечивают доступ не ко всей сети, а только к тем серверам, с которых загружаются классы, используемые интерпретируемым апплетом. Кроме того, имеется возможность организовать барьер, предотвращающий доступ к ресурсам компьютера пользователя апплета со стороны сети.

Отметим, что если третий и четвертый из вышеперечисленных уровней нетрудно организовать и в системах программирования других языков, то организация первых двух уровней возможна только в системе Java, так как эти уровни основаны на свойствах языка и системы программирования Java, которых нет в других языках.

Проблема эффективности Java-программ и оптимизирующие компиляторы языка Java. Здесь под эффективностью имеется в виду скорость выполнения Java-программ. Поскольку Java-программы интерпретируются на JavaVM, они, естественно, выполняются медленнее, чем программы, разработанные с помощью оптимизирующих компиляторов C/C++.

В тех редких случаях, когда возникает потребность увеличить скорость выполнения апплета, можно воспользоваться динамическим компилятором Java (just-in-time compiler), который работает параллельно с интерпретацией апплета и переводит байткод в объектный код компьютера, на котором выполняется апплет. Этот объектный код оформляется как обычная подпрограмма и (после загрузки) вызывается из JavaVM (при программировании на Java разрешается использовать подпрограммы, выполняемые в объектном коде компьютера, на котором работает JavaVM (native subroutines); эти подпрограммы считаются подпрограммами самой JavaVM).

Подробнее о технологии Java и использовании ее в гипертекстовых документах можно узнать из других источников [11, 14, 15, 16, 17, 18].

1.2.2. ActiveX-технология

Свое достаточно позднее вступление в борьбу за Web компания Microsoft сопроводила завесой громких заявлений. Многие из разработчиков, увлекшись языком Java фирмы Sun и платформой Netscape Navigator, восприняли обещания Microsoft как обычный рекламный трюк. И оказались неправы. Многие из компонентов платформы ActiveX уже получили путевку в жизнь и стали незаменимыми при решении задач определенного класса, прежде всего для интрасетей.

По своей сути платформа ActiveX - это адаптация существующих технологий Microsoft применительно к Web [19, 20, 21]. Данная концепция базируется на механизмах OLE и COM - проверенных временем и ставших стандартами разработок для Windows. VBScript - язык программирования ActiveX-объектов на страницах Web - представляет собой диалект языка Basic приоритетного для Microsoft с момента ее основания. Имеющиеся ныне инструменты для подготовки Web-страниц на базе технологии ActiveX явно обязаны Microsoft Visual Basic. В целом адаптация прошла удачно.

Что же это такое на самом деле? Наиболее простой ответ и, как оказывается, наиболее точный: "управляющий элемент ActiveX" - это просто новый маркетинговый термин вместо "управляющего элемента OLE" (называвшегося прежде "управляющим элементом OCX"). Любой готовый или созданный вами управляющий

элемент OLE - это уже ActiveX-элемент и может использоваться в оснащенных средствами для работы с ним программах. Однако подобные OLE-элементы составляют лишь часть всего многообразия управляющих элементов ActiveX и совсем не обязательно устроят разработчиков для Web.

Типичный недостаток существующих OLE-элементов - их размеры слишком велики. Это обусловлено сложностью структуры OLE-интерфейсов при полной их реализации управляющими элементами, а также тем фактом, что при подготовке в Microsoft библиотек, используемых для генерации управляющих элементов, размеры их не оптимизировались (мягко говоря). Если в системе пользователя какой-то из этих элементов отсутствует, приходится загружать его через Internet, следовательно, размер у управляющих элементов Web страниц должен быть как можно меньше. Некоторые из существующих OLE-элементов удовлетворяют этому критерию. Для устранения названного недостатка в Microsoft предложили два следующих варианта.

Первый вариант: упростить определение ActiveX-элементов в части, касающейся реализуемых интерфейсов. С технической точки зрения ActiveX-элемент - это некоторый COM-объект, через основной OLE-интерфейс которого, IUnknown, организуется доступ к остальным интерфейсам данного объекта. Благодаря такой схеме разработчики могут обеспечивать лишь те OLE-интерфейсы, которые действительно необходимы для текущей задачи, не нарушая при этом требований существующей спецификации. Второй вариант: "перекроить" базовые средства, используемые для подготовки ActiveX-элементов.

Вероятно, многие управляющие элементы ActiveX, т. е. OLE, хорошо знакомы разработчикам, но, наверное, среди них есть и такие, которым необходимы пояснения. По существу с помощью ActiveX-элементов программист создает некоторый высокоуровневый, пригодный для многократного использования объект с конкретной полезной функцией. Затем этот элемент может быть передан (или продан) другому программисту, которому пригодится как некий "кирпичик" в его программном инструменте, обладающем средствами для взаимодействия с ActiveX-элементами. (Здесь речь идет о предназначенном для Windows большинстве основных

инструментов для Си/Си++, Basic, Pascal и многих других языков.) В роли ActiveX-элементов может быть все что угодно - от обычной кнопки до полнофункциональной электронной таблицы. В продаже имеются тысячи таких элементов от разных поставщиков. Их богатые функциональные возможности и многообразие - отдельное, наиболее важное достоинство платформы ActiveX.

ActiveX-элементы - это гораздо более мощное средство, чем HTML формы. Управляющие элементы добавляются к HTML с помощью команды <OBJECT>. Для внутренних обращений к объекту задают его идентификатор ID и параметр CLSID - 128-разрядный глобальный уникальный идентификатор (GUID) данного OLE элемента. При вызове ActiveX элемента из VBScript система просматривает системный Реестр Windows, чтобы выяснить, зарегистрирован ли он там. Если нет, система использует атрибут CODEBASE (из команды <OBJECT>), в котором указан адрес сервера с требующимся элементом. После его загрузки и инсталляции, проводимых некой встроенной системной службой, данный элемент готов к работе.

Для построения ActiveX компонентов существует несколько технологий.

Любой OLE-элемент - это одновременно и ActiveX-элемент (хотя обратное справедливо не всегда). Поэтому, если вам приходилось работать с библиотекой Microsoft Foundation Classes (MFC) и комплектом инструментов Control Developer's Kit (CDK), считайте, что один способ создания собственного ActiveX-элемента вам уже знаком. Подготовленный средствами MFC OCX-файл может быть сравнительно небольшим, но всецело зависит от наличия DLL-библиотеки для обслуживания MFC-функций, размер последней версии которой достигает почти одного мегабайта. Поэтому если пользователь работает с такой Web страницей, где используется совершенно новый ActiveX-элемент, то его загрузка по сети потребует какого-то времени, т. е. возникает небольшая задержка. Мы, наверное, уже привыкли к такого рода ожиданиям, но в данном случае время загрузки может превысить и 5 мин, например, если DLL-библиотека размером в 1 Мбайт загружает через 28,8-кбит/с модем. Естественно, пользователь будет обвинять автора данной Web страницы во всех

грехах, тогда как последний отнесет их на счет разработчика этого элемента. Вот почему большинство программистов, занимающихся подготовкой ActiveX-элементов, заинтересованы в замене CDK другими средствами.

Один из вариантов замены CDK - библиотека ActiveX Template Library (ATL), разработанная группой специалистов из Microsoft, ответственной за создание языка Visual C++. ATL работает в рамках среды MFC, тем не менее размер сформированных на ее основе управляющих элементов существенно меньше. Для тех, у кого есть опыт программирования на базе MFC, вполне разумным решением станет, вероятно, использование ATL.

Однако для создания действительно компактных элементов придется отказаться от MFC и даже от библиотек рабочих программ C. Обратившись к Web-узлу Microsoft и ознакомившись с примерами и документацией к инструментарию ActiveX Controls Framework, вы поймете, как это сделать. Для работы потребуются знания об основах механизмов OLE и ActiveX, однако на вполне достижимом для рядовых программистов уровне.
К сожалению, этого не скажешь об еще одном варианте для решения этой проблемы - системе разработки эталонных управляющих элементов, носящей название BaseCtl. Ее созданием занималась группа специалистов Microsoft, в свое время давшая жизнь Internet Explorer. Этот инструментарий входит в пакет ActiveX Development Kit. Отрадно то, что подготовленные на базе BaseCtl элементы максимально компактны. Но не обольщайтесь: для их создания потребуется глубокое понимание работы механизмов COM, OLE и ActiveX.

Технология ActiveX заслуживает пристального внимания, но ей еще предстоит трудная борьба за Web. Средства для ее использования имеются пока только в браузере Internet Explorer, который, в свою очередь, ограничивается лишь средой Windows - по крайней мере на текущий момент. (По словам представителей Microsoft, в будущем технология ActiveX будет подготовлена и для платформ Macintosh и UNIX.) Средства для работы с управляющими элементами ActiveX, может быть, найдут пристанище и в браузерах других поставщиков, однако, как сообщает Netscape

браузер Netscape Navigator останется в стороне от этой технологии.

Компонентная структура ActiveX-технологии представлена на рис. 5.

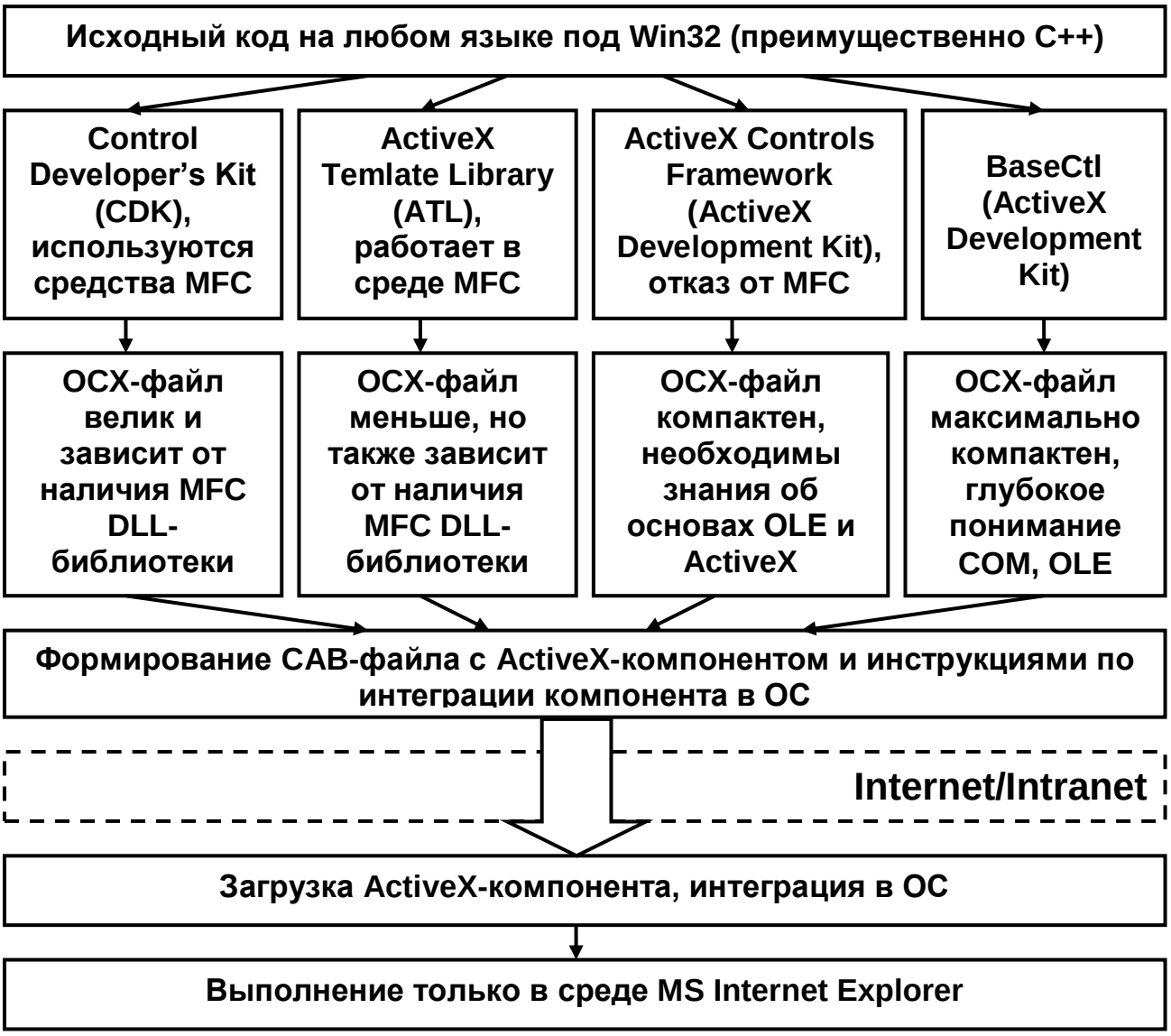


Рис. 5. Microsoft ActiveX-технология

1.2.3. Сравнительный анализ технологий

Практика использования Java-аплетов и управляющих элементов ActiveX для разработки Web-приложений пока еще редкость, но рост популярности систем на основе тонких клиентов придает ей дополнительный импульс. Развертывание этих «облегченных» программ упрощает решение традиционных задач распространения и контроля версий ПО, поскольку гарантирует, что ваши

пользователи всегда будут работать с самой «свежей» версией программного кода.

Технологии Java и ActiveX вполне пригодны для создания интерфейсов или приложений, специально предназначенных для применения в корпоративных сетях (интрасетях или сетях extranet) [21]. Если же ключевой момент стратегии развития сети состоит в обеспечении Web-функциональности, то использование Java-аплетов или управляющих элементов ActiveX напрашивается само собой. При этом, если аплеты или управляющие элементы являются составной частью сложного и интенсивно используемого корпоративного приложения, то следует тщательно выбирать сервер Web-приложений.

Далее рассматриваются некоторые проблемы, возникающие при разработке и развертывании этих загружаемых из сети небольших исполняемых модулей.

Java против ActiveX. Сначала определимся: когда лучше всего использовать Java-аплеты, а когда предпочесть им управляющие элементы ActiveX [20]?

Если требуется поддержка нескольких платформ сразу, то, безусловно, лучше всего выбрать Java. Элементы ActiveX тоже поддерживают некоторые не-Windows платформы, например такие, как Macintosh фирмы Apple Computers, однако Java обеспечивает куда более широкую поддержку клиентов и ОС. Кроме того, если необходим доступ извне к вашим ресурсам с помощью различных Web-браузеров (как, например, в сети типа extranet), то опять же Java имеет преимущества перед ActiveX и с точки зрения безопасности, и с точки зрения использования, поскольку без установки дополнительных модулей (plug-ins), например таких, как ScriptAccess компании NCompass Labs, элементы ActiveX не поддерживаются браузерами Netscape. Кроме того, в технологии Java благодаря использованию файлов Resource Bundles значительно легче локализовывать программы. Разумеется, в Java пока не полностью реализован принцип «пишется однажды — работает везде» (write once, run anywhere), тем не менее эта технология действительно лучше переносится на другие платформы, особенно по сравнению с ActiveX.

Элементы ActiveX больше всего подходят для контролируемых сред, например для приложений,

работающих в интрасетях. Для создания цифровой подписи программного кода элементов ActiveX с целью обеспечения дальнейшей аутентификации служит фирменный метод Microsoft — Authenticode. Это, впрочем, не гарантирует их безвредность для работы хост-системы в отличие от Java-аплетов, работающих в так называемой «песочнице» (sandbox). Элемент ActiveX имеет доступ к файловой системе хоста и может свободно распоряжаться распределением оперативной памяти.

Если вы работаете только в средах, полностью базирующихся на ОС Windows, а вашим любимым браузером является Microsoft Internet Explorer, то технология ActiveX представляется вполне логичным выбором. Ее элементы реализуют модель одноразовой загрузки из сети, что сокращает время выполнения приложений при последующих обращениях к уже загруженному элементу (в версии 1.1 спецификации Java апплеты могут функционировать подобным образом, но не со всеми браузерами и только при условии наличия цифровых подписей в формате X.509). Если вы привыкли использовать в своих приложениях заказные управляющие элементы OCX (OLE Custom Control), написанные на языке Си++ или других языках программирования, то технология ActiveX позволит воспользоваться некоторыми из унаследованных программ.

Следует учесть, что свойства технологии могут в одном случае привести к возникновению проблем с безопасностью, а в другом — обернуться сущим благом. Например, простота доступа к файловой системе и прочим локальным ресурсам машины существенно облегчает разработку приложений для интрасетей. Другим положительным свойством является возможность воспользоваться самыми разными языками программирования, включая Си++, Delphi, PowerBuilder, Visual Basic и даже Java, для написания исходного кода элемента ActiveX.

Принятие решения. Прежде чем создавать элементы ActiveX, следует выбрать одну из двух библиотек: Microsoft Foundation Class (MFC) или ActiveX Template Library (ATL). При работе с MFC управляющие элементы получаются более громоздкими и требуют установки на локальной машине пользователя корректной версии динамически подсоединяемой библиотеки MFC, иначе при

работе с этими элементами могут возникнуть проблемы. Преимущество такого подхода заключается в том, что разработчики смогут уделять больше времени самому объекту программирования и меньше заботиться о разработке интерфейса.

Библиотека ATL труднее в изучении, в ней меньше ресурсов и примеров, чем в MFC. Однако она оптимизирована для работы с Интернет, что позволяет создавать менее тяжеловесные управляющие элементы, не требующие инсталляции MFC на каждой хост-системе.

Одной из главных трудностей при разработке Java-приложений является принятие решения, для какой из версий виртуальной Java-машины (Java Virtual Machine — JVM) следует писать программу. Если вы хотите, чтобы приложение работало на самых разных клиентских системах, то «общим знаменателем» для них станет JVM версии 1.0.2. Однако у нее есть целый ряд серьезных недостатков, в том числе: примитивные элементы управления графического интерфейса пользователя (GUI controls), минимальное взаимодействие с базами данных, недостаточная степень детализации модели безопасности (например, операции файлового ввода-вывода запрещены полностью). Кроме того, с ее помощью нельзя выполнять некоторые самые простые задания, например вывод на печать.

JDK (Java Development Kit — набор средств разработки Java-приложений) версии 1.1.7 и недавно вышедшей версии 1.2 были значительно усовершенствованы и существенно обновлен набор их функций [18]. В частности, в них появились: возможность использования апплетов с цифровой подписью (при определенных условиях это позволяет им хотя бы частично выйти за ограничивающие рамки «песочницы»); компонентная модель JavaBeans, файлы типа JAR, позволяющие «упаковывать» все необходимые классы в один загружаемый файл; мощная модель обработки событий и значительно усовершенствованный набор компонентов графического пользовательского интерфейса (Abstract Windowing Toolkit — AWT).

JDK версии 1.2 [18] тоже предлагает очень мощную и многоуровневую модель безопасности, основанную на правилах разграничения доступа. В ней предусмотрены: графическая консоль пользователя для создания и поддержки базы данных правил; средство для управления

заверенными цифровыми сертификатами X.509 версии 3 и ключами шифрования; компонент JARsigner, предназначенный для снабжения JAR-файлов цифровыми подписями и их подтверждения. Начиная с JDK версии 1.1, в Java реализован механизм JDBC (Java Database Connectivity), который благодаря поддержке совместимости JDBC-to-ODBC гарантирует доступ Java-приложений к самым разнообразным данным.

Встраиваемые модули. Серьезной проблемой для спецификации Java версии 1.1 является отсутствие полной поддержки ее как со стороны Netscape Navigator, так и Microsoft Internet Explorer. Наилучший способ справиться с этой проблемой — это использовать встраиваемый модуль (plug-in) для поддержки Java, который вместо поставляемой вместе с браузерами Netscape Navigator и IE версий 3.x и 4.x виртуальной Java-машины позволяет воспользоваться оперативными средствами управления Sun RTE (Run-Time Environment) версии 1.1 или 1.2. Этот модуль делает возможным не только полное использование средств Java версии 1.1 и 1.2, но и гарантирует их совместимость, что еще важнее. Модуль plug-in для Java 1.2 предоставляет доступ к интерфейсу API, используемому для получения электронной подписи и регистрации аплетов, а также смягчает накладываемые «песочницей» ограничения и кэширует JAR-файлы на локальном диске.

Встраиваемые модули удобно применять в интрасетях, однако они могут оказаться неподходящими для приложений, предназначенных для сетей extranet, особенно если вы не в состоянии контролировать тип и версию браузера конечного пользователя. В этом случае придется обратиться к Java версии 1.0.2.

Развертывание аплетов и управляющих элементов. Контроль корректности версий управляющих элементов и аплетов по идее не представляет особой сложности. Любая стандартная программа контроля версий, например такая, как Microsoft Visual SourceSafe, должна уметь работать с файлами JAR, классами Java и управляющими элементами ActiveX. Для нужд же небольших команд разработчиков отлично подойдет JavaSafe — программа контроля программного кода Java от фирмы Sun. Однако непосредственное развертывание и использование апплетов и управляющих элементов — это совсем другая

история; чтобы эти процессы протекали гладко, необходимо выполнить ряд действий.

Для Java-аплетов, подлинность происхождения которых проверяется с помощью цифровой подписи, по порядку необходимо выполнить следующее:

1. Скомпилировать аплет;
2. Создать файл JAR;
3. Сгенерировать ключи;
4. Подписать файл JAR;
5. Произвести экспорт сертификата с открытым ключом;
6. Произвести импорт этого сертификата как доверенного;
7. Создать файл правил разграничения доступа (policy file);
8. Запустить аплет.

Для ActiveX каждый управляющий элемент должен быть подписан и помечен как безопасный или небезопасный [20].

Существует эмпирическое правило: если элемент ActiveX осуществляет операции чтения или записи на диск, либо чтения или записи за пределами выделенной ему оперативной памяти (чего не может делать Java-апплет, помещенный в «песочницу»), значит, он небезопасен.

Многокомпонентные элементы ActiveX, которые работают с браузером IE 4.0 фирмы Microsoft, лучше всего размещать таким образом, чтобы они обеспечивали требования открытого стандарта описания ПО (Open Software Description — OSD), являющегося подмножеством словаря XML. Браузер IE 4.0 позволяет целиком встраивать OSD-файлы в файлы CAB, в которых содержатся также INF-файлы элементов ActiveX. Помимо этого стандарт OSD обеспечивает локализацию и поддержку различных платформ.

Следует убедиться и в том, что у всех пользователей установлена одна и та же версия JVM или одна и та же версия библиотеки MFC (для тех элементов ActiveX, что требуют использования MFC, а не ATL). Работа с неверной версией JVM или MFC может стать причиной возникновения проблем несовместимости.

Для Java наилучшим способом обеспечения совместимости является использование соответствующего модуля Java plug-in. Короче говоря, необходимо

унифицировать все версии браузеров — это единственный способ, который позволит программисту сосредоточиться на одной конкретной версии JVM и учесть все присущие ей характерные особенности. И наконец, не стоит забывать, что браузеры должны быть соответствующим образом сконфигурированы, т. е. в них должны загружаться и активизироваться Java-аплеты и управляющие элементы ActiveX, иначе все хитроумные разработки будут недоступны для пользователей.

1.3. Существующие системы поддержки изучения цифровой схемотехники

1.3.1. Основные проблемы, возникающие при изучении цифровой схемотехники

Цифровая схемотехника – это одна из дисциплин, которой недостает наглядного представления изучаемых процессов [41, 42, 48, 55, 56]. В этой дисциплине при анализе какого-либо цифрового устройства пользуются моделью так называемого "черного ящика", при которой реальному наблюдению доступны лишь значения сигналов на входах и выходах устройства - некоего "черного ящика"; процессы же, происходящие внутри него - смена логических состояний составляющих его компонентов, изменение значений потенциалов в электрических цепях – требуют абстрактного представления. Полноценное модельное представление этих процессов возможно только с помощью ЭВМ.

Необходимо отметить также, что традиционные способы изучения предмета цифровой схемотехники предполагают этапы объяснения учебного материала и закрепления знаний с помощью различных элементов организации учебной деятельности. Является полезным объединение в одном роде деятельности этапов объяснения и закрепления материала. В этом случае уменьшается время, необходимое для переключения с одного рода деятельности на другой, связанное с различной формой представления материала, например, на классной доске и лабораторных стендах. Также становится возможным более гибко сочетать этапы объяснения и закрепления.

1.3.2. Учебная система «Последовательностные цифровые устройства»

Учебная система "Последовательностные цифровые устройства" создана в Хакасском государственном университете [57] для изучения одноименного раздела цифровой электроники. Она, сохраняя все вышеперечисленные преимущества компьютерного метода, объединяет в едином подходе процессы объяснения и закрепления знаний, а также реализует эффективный механизм контроля уровня усвоения полученных знаний.

Отличительные качества программы связаны с ориентированием ее на компьютерную поддержку занятий, в том числе на моделирование процессов, происходящих в цифровых устройствах и не поддающихся реальному наблюдению. Благодаря этим качествам повышаются наглядность, полнота и эффективность получаемых знаний. Кроме того, программа позволяет отрабатывать специальные для этой дисциплины навыки работы, как-то: анализ и построение временных диаграмм и таблиц переключений цифровых устройств; а также оперативно контролировать качество закрепления этих навыков.

Программа "Последовательностные цифровые устройства" использует знания о логических элементах для изучения устройств, построенных на ее основе. Это обусловлено принципом преемственности знаний: изучение последовательностных цифровых устройств возможно только после изучения логических элементов.

Работа с программой построена на обучающих фрагментах единого образца по различным видам цифровых устройств из числа триггеров, регистров и счетчиков. Программа содержит в себе 24 таких фрагмента. Обучение строится на последовательной основе: сначала обучаемым даются основные сведения по изучаемому устройству, затем организуется закрепление знаний, после чего возможен переход к его контролю. Работа с программой облегчена системой контекстных подсказок, имеющихся в каждом режиме работы, и обширным текстовым содержанием обучающих блоков: суммарный объем текстовой информации обучающей части составляет 45 килобайт.

Поскольку триггеры входят в состав регистров и счетчиков, рассмотрение счетчиков и регистров с помощью программы является целесообразным после

изучения триггеров именно с использованием этой программы. Это тем более важно, поскольку в программе учитывается преемственность знаний и происходит постепенное усложнение содержания обучения, а также весьма важным является постепенное формирование навыков работы с программой при изучении последовательностных устройств.

Программа на этапе объяснения избавляет преподавателя от нерационального использования учебного времени: весьма приблизительного характера построений схем цифровых устройств, медленного заполнения таблиц переключений и временных диаграмм. Кроме того, в программу включена возможность наглядного наблюдения изменения состояний отдельных элементов схем цифровых устройств при различной комбинации значений входных сигналов.

Целью этого этапа является описание и объяснение функциональных назначений устройств; построение временных диаграмм и заполнение таблиц переключений значениями выходных сигналов, получаемыми подачей различных комбинаций сигналов на входы устройства.

Закрепление знаний сопровождается формированием навыков работы с таблицами переключений сигналов и временными диаграммами.

На этом этапе обучаемый на практике проверяет истинность своих представлений о работе устройства. Для этого он может самостоятельно задавать значения входных сигналов устройства и давать компьютеру команду на их обработку. Программа при этом устанавливает устройство в новое состояние в соответствии с поданными сигналами, а обучаемый видит насколько его представления о работе устройства совпали с истинным результатом.

Контроль знаний заключается в том, что компьютер предлагает обучаемому комбинацию входных значений, а тот должен определить значения сигналов на выходах устройства.

При этом программа предоставляет преподавателю удобный механизм контроля знаний обучаемого по данному разделу. Для обеспечения большей оперативности оценки выставляются не только по частным темам, но и по обобщающим разделам и подразделам видов цифровых устройств. Имеется возможность корректировки требований к обучаемым при

выставлении оценок. Ввиду большого объема изучаемых тем в программе предусмотрен отбор разделов при контроле знаний.

На практике данное программное средство можно применять при изучении цифровой электроники в вузах и при ведении факультативных занятий по прикладной физике в старших классах общеобразовательных учреждений.

Работа с программой возможна по следующим направлениям:

- использование ее для поддержки лекционных занятий, где благодаря применению динамической графики, быстрому произвольному построению таблиц и временных диаграмм заметно повышается качество и количество передаваемых преподавателем знаний, и высвобождается учебное время для более глубокого объяснения материала;
- большое количество комментариев и обширное текстовое содержание программы делает ее применимой и для самостоятельного изучения обучаемыми последовательностных устройств на основе имеющихся у них знаний о цифровых логических элементах;
- программу можно использовать для организации лабораторного практикума наряду с работой на традиционных лабораторных стендах.

При создании программы в нее была заложена единая для всех цифровых устройств компьютерная модель. Это позволяет, изменив содержание обучения, предложить программу для изучения других цифровых устройств. При этом алгоритмическая структура программы остается прежней.

Модель работает на базе библиотеки данных о структуре цифровых устройств. Структура любого устройства описывается заданием всех связей с выхода на вход между составляющими элементами и типом используемых элементов. В качестве составляющих элементов могут использоваться либо логические элементы, закон функционирования которых задан таблицами истинности в самой программе, либо другими устройствами, уже описанными в библиотеке. Демонстрация функционирования любого устройства из библиотеки при определенной комбинации сигналов на

входах устройства осуществляется поэтапной по времени передачей значений логических уровней с выхода предыдущего элемента на вход следующего и анализом изменения состояния составляющих элементов. При этом на экране компьютера синхронно отражается значение потенциалов соединительных проводников и составляющих элементов, графическое изображение которых также берется из библиотеки данных. Для анализа изменения состояния элемента, составляющего рассматриваемое устройство, используется рекурсивный алгоритм, при котором анализ каждого элемента строится, исходя из его структуры.

Модель функционирования цифровых устройств предназначена для описания устройств, работающих по двухстабильной системе. При создании модели были приняты некоторые упрощающие допущения. В частности, для логических элементов и устройств принято одинаковое время задержки сигнала со входа на выход, в результате чего модель не рассматривает эффекты мультивибрации триггеров и др. Еще одним упрощением является принятие одинакового для всех устройств коэффициента разветвления по выходу, совпадающего с коэффициентом большинства логических элементов серии ТТЛ.

Минимальная конфигурация компьютера, необходимая для работы программы: процессор 80286, объем ОЗУ – 1 Мбайт, видеоадаптер - EGA 640x350. Требования к базовому программному обеспечению: операционная система DOS 6.x и выше с русским знакогенератором, Windows3.x или Windows9x.

1.3.3. Учебная система «Проектирование цифровых устройств на вентильных матрицах»

Система разработана в Пензенском государственном университете [52, 58] и предназначена для проектирования цифровых устройств на вентильных матрицах на функционально-логическом уровне. Вопросы топологического проектирования в системе не решаются. Система позволяет также контролировать знания студентов по схемотехническим дисциплинам.

Система обеспечивает:

- разработку схем средней сложности на ТТЛ или ЭСЛ элементах;

- задание входных сигналов;
- моделирование работы разработанных схем;
- получение результатов в виде временных диаграмм;
- контроль знаний студентов на относительно простых схемах с выдачей результатов контроля и возможностью накопления статистики результатов контроля.

Технические характеристики:

- режим работы - диалоговый;
- размер вентильной матрицы - 2000 вентилей И-НЕ;
- количество входных сигналов - 16;
- количество точек контроля - 16.

Система может использоваться в лабораторных занятиях и курсовом проектировании при проведении схемотехнических дисциплин.

Система имеет меньшие возможности, чем профессиональные системы проектирования. Однако, засчет ограничения возможностей в части проектирования схем и режимов моделирования удалось сделать систему более простой для освоения студентами и повысить скорость работы системы.

Система ориентирована на работу с библиотекой элементов: логических схем, триггеров, регистров и других, в основном аналогов микросхем серий К155 (К155ТВ1, К155ИР1, К155ИЕ7, К155ИМ3 и других) и К500 (К500ЛМ105, К500ТМ131, К500ИР141, К500ИЕ136 и др.). Библиотека элементов может быть расширена путем разработки и добавления новых элементов.

Система функционирует на ЭВМ, совместимых с IBM PC XT/AT в операционной среде MS DOS. В состав системы входят 2 программы, размером 138 и 175 Кбайт, файл библиотеки и файл подсказки общим объемом 30 Кбайт.

1.3.4. Проектирование цифровых устройств в системе HLCCAD

Система высокоуровневого проектирования чипов HLCCAD (High Level Chip Computer Aided Design) разработана и продолжает усовершенствоваться в Гомельском государственном университете им. Ф.Скорины [59]. HLCCAD – мощная система разработки цифровых устройств, в том числе микропроцессорных (с

поддержкой мультипроцессорности) и программно-аппаратных, работающих под управлением ПК [43, 44, 45]. HLCCAD позволяет производить редактирование, моделирование и анализ работы устройств, а также создавать выходные описания на синтезируемом VHDL для переноса в системы проектирования более низкого уровня с целью окончательного синтеза чипов.

Одним из главных достоинств системы является возможность представления устройств на разных уровнях абстракций для иерархического проектирования. При этом разработка может вестись как сверху вниз (т.е. от более общего уровня к детальному), так и снизу вверх. (когда более сложные компоненты “собираются” из уже готовых).

Каждый компонент в HLCCAD представлен корпусом (т.е. УГО, которым этот компонент обозначается на схемах), а также внутренней схемой либо DLL-моделью. В последнем случае описание логики работы компонента осуществляется на языке высокого уровня (Delphi, C++ и др.). Предусмотрена возможность создавать библиотеки компонентов. Наиболее распространенные элементы (логика, память, регистры, дешифраторы, генераторы и пр.) объединены в библиотеку стандартных компонентов. Эта библиотека является основой для построения схем более сложных устройств.

Система HLCCAD работает в среде Windows 95/98/NT и обладает весьма удобным интерфейсом в лучших традициях стиля Borland Delphi. При входе в систему пользователь обнаруживает два окна – главное и инспектор проектов. В главном окне сосредоточены главное меню и иконки основных команд. Окно инспектора проектов отображает подключенные проекты и их иерархию.

Для просмотра и изменения схем предусмотрено окно редактора, которое открывается из окна инспектора проектов. Окно редактора имеет два режима – редактирование корпусов и редактирование схем. Редактор корпуса предназначен для редактирования УГО компонента. Здесь же задаются различные параметры – количество и типы контактов корпуса, тип внутреннего представления компонента (схема или DLL-модель), величина временной задержки прохождения сигналов через контакты корпуса, пути к файлам с прошивкой для элементов памяти.

Редактор схем позволяет создавать внутреннюю структуру устройства. Чтобы добавить в схему элемент достаточно просто перетащить его из окна инспектора проектов в окно редактора. Редактор позволяет задавать связи между элементами путем рисования линий и шин. Также поддерживаются “невидимые” линии – т.е. для выбранных контактов задаются только имена связей, а контакты с одинаковыми именами связей соединяются невидимой шиной автоматически. Это существенно повышает скорость редактирования и удобочитаемость схем. Простота и удобство редактирования достигаются также за счет таких полезных операций как многоуровневый откат, копирование и вставка элементов и участков схем через буфер, режим «резиновых» линий (т.е. перемещение элемента по схеме без нарушения связей с другими элементами).

После редактирования можно приступать к этапам моделирования и анализа. Система выполняет моделирование с учетом временных задержек и с использованием файлов тестовых последовательностей. При этом в файле тестов содержатся команды, позволяющие подавать входные воздействия в определенные моменты времени и контролировать состояние контактов устройств (проверять на совпадение значений на контакте с тестовым).

Возможно пошаговое моделирование или моделирование в фоновом режиме. При этом можно анализировать результаты уже смоделированного диапазона времени (открывать окна для анализа схем, просмотра истории значений на контактах и т.д.). При пошаговом моделировании можно установить значения на входах схемы и продолжить моделирование с новыми значениями.

HLCCAD предоставляет пользователю мощные средства анализа результатов моделирования. Предусмотрена возможность наблюдать историю смены значений на контактах элементов прямо на схемах, переходя из одного уровня иерархии в другой, или в специальном окне истории значений, в котором собрана информация о том какие значения в какие моменты времени появлялись на контактах. Удобным средством анализа является окно временных диаграмм. Выбор контактов для окон истории значений и временных диаграмм осуществляется перетаскиванием элементов или контактов из окон схемы

и дерева модели. При этом может использоваться механизм фильтров – в окна попадают контакты, прошедшие через фильтр (т.е. удовлетворяющие заданным пользователем условиям). Для элементов памяти можно открывать окно дампа и просматривать содержимое. Для всех значений пользователь имеет возможность задавать систему счисления, а также использовать перечисляемые типы.

Если в устройстве применяются микропроцессоры, для них можно открывать окно дизассемблера и следить за ходом выполнения программы одновременно с анализом реакции схемы.

1.3.5. Micro-CAP и Micro-Logic - интегрированные системы моделирования логических и цифровых схем

Системы Micro-CAP ориентированы на моделирование аналоговых электронных схем с их полной детализацией [38, 27]. Исключением являются, пожалуй, лишь схемы на интегральных операционных усилителях, в которых последние представлены своими упрощенными макромоделями. Впрочем, в системы введена возможность задания макросов (подсхем) с помощью которых можно моделировать даже логические и цифровые схемы. Однако время моделирования в этом случае значительно, поскольку в ходе моделирования составляются и решаются большие системы уравнений, описывающих работу схем.

С появлением микроэлектроники возник и интенсивно развивается новый класс схем - интегральные логические и цифровые микросхемы. Для пользователя они являются неким "черным ящиком" с известным и довольно простым поведением. Полная принципиальная схема такой микросхемы, как правило, пользователю неизвестна и в справочной литературе даже не приводится. Из логических и цифровых микросхем могут создаваться сложные электронные системы - от наручных электронных часов и калькуляторов до вычислительных машин, сложнейших систем управления аэропортами и космодромами.

В корне меняется подход к моделированию таких схем - вместо детального моделирования (на уровне учета каждого компонента) используется макромоделирование, при котором схема описывается как многополюсник с известными логическими функциями.

Основной задачей моделирования является получение временных логических диаграмм работы устройства в том виде, который выдают специальные испытательные устройства - логические анализаторы.

Для реализации этих возможностей фирма Spectrum Software наряду с системами Micro-CAP создала интегрированные системы моделирования логических и цифровых схем серии Micro-LOGIC. Эти системы позволяют конструировать логические или цифровые схемы из набора компонентов в виде стандартных интегральных логических и цифровых микросхем. Можно описывать и свои нестандартные логические схемы.

Micro-LOGIC позволяет (на уровне моделирования, разумеется) подключать к заданным точкам схем тактовые и испытательные сигналы и наблюдать временные диаграммы работы моделируемой схемы как в статике, так и в динамике.

Благодаря реализации принципов макромоделирования системы Micro-LOGIC позволяют быстро моделировать довольно сложные логические и цифровые схемы, которые явно "не по зубам" системам Micro-CAP. В то же время пользовательский интерфейс у систем Micro-LOGIC очень похож на используемый в системах Micro-CAP III. Сохранена и ориентация на ПК класса IBM PC.

Система Micro-LOGIC имеет заметно более узкое функциональное назначение, чем у систем Micro-CAP. Единственный тип компонентов, с которыми работает система (не считая проводниковых линий связи) - это логические схемы. В моделируемую схему нельзя включать ни транзисторы, ни даже конденсаторы или резисторы. Поэтому систему нельзя использовать для моделирования нестандартных схем на логических элементах - например мультивибраторов или формирователей импульсов. Для этого надо использовать системы класса Micro-CAP. Система Micro-LOGIC ориентирована на моделирование чисто логических и цифровых устройств.

Это, бесспорно, существенное ограничение. В тоже время надо учитывать, что объемы применения логических микросхем сейчас намного превосходят таковые для аналоговых микросхем. Поэтому сфера применения системы Micro-LOGIC чрезвычайно широка и эта система может оказать неоценимую пользу при разработке различных логических и цифровых устройств.

В систему заложена возможность ее расширяемости путем добавления новых компонентов и редактирования их параметров и графических изображений (образов).

Совместно системы Micro-CAP и Micro-LOGIC позволяют моделировать огромное число самых разнообразных электронных схем и устройств. Их применение способствует разработке новых схемотехнических решений и повышению эффективности работы электронных устройств (в том числе надежности, долговечности, технологичности и быстродействия). Можно сказать, что эти системы взаимно дополняют друг друга и созданы как единый комплекс схемотехнического моделирования с удобным графическим и многооконным пользовательским интерфейсом.

1.3.6. САПР P-CAD: назначение и характеристика основных программ

P-CAD является промышленной системой автоматизированного проектирования радиоэлектронной аппаратуры [36, 37]. Структурно P-CAD представляет собой интегрированный набор специализированных программных пакетов, работающих в интерактивном режиме и взаимодействующих друг с другом лишь через общие базы данных.

Система P-CAD 2000 выполняет полный цикл проектирования печатных плат, а именно:

- графический ввод электрических схем,
- упаковку схемы на печатную плату,
- интерактивное размещение компонентов,
- ручную, интерактивную и/или автоматическую трассировку проводников,
- контроль ошибок в схеме и печатной плате и выпуск документации,
- анализ целостности сигналов.

Функциональные возможности P-CAD 2000:

- Задание форматов перечней компонентов и других отчетов (PCB, Schematic). Пользователям предоставляется возможность задавать перечень полей отчета и порядок их следования,
- Максимальное количество секций в компоненте до 5 000. Максимальное количество выводов компонента – 10000,

- Измерительный инструмент Miter Tool распознает не только прямые, но и произвольные углы, принимая их в качестве точек «привязки» при выполнении измерений,
- Электронная документация имеет механизм простого доступа к информации о компонентах (PCB, Schematic). Введена папка со списками URL's (адресов в Internet) всех ведущих фирм, производителей полупроводниковых компонентов,
- Имеется механизм переноса изменений печатной платы на схему и наоборот (Engineering Change Order, ECO),
- Поддерживается как английская, так и метрическая система единиц,
- Применение 32-разрядной арифметики обеспечивает дискретность измерения линейных размеров 0,1 мил в английской системе (1 мил = 0,001 дюйма) и 0,001 мм в метрической системе, угловых размеров 0,1 град. и возможность изменения системы единиц на любой стадии работы с проектом без потери точности. Напомним, что в P-CAD для DOS система единиц устанавливалась до начала работы с проектом и в дальнейшем не могла быть изменена,
- Поддержка текстовых форматов описания баз данных DXF и PDIF позволяет обмениваться информацией с такими распространенными пакетами, как AutoCAD, OrCAD, Viewlogic, конечно, P-CAD для DOS и др,
- Многошаговый откат вперед и назад (Undo и Redo),
- Автоматическое размещение компонентов на плате и эффективная автоматическая трассировка проводников реализованы в поставляем отдельно пакете SPECCTRA 9.x фирмы Cadence Design Systems (ранее Cooper&Chyan Technology),
- P-CAD 2000 поставляется с большой библиотекой современных импортных ЭРЭ, которую можно пополнить библиотеками отечественной элементной базы, в частности, импортированными из P-CAD для DOS,

- P-CAD 2000 (ACCEL EDA) и SPECCTRA 9.x устанавливаются на ПК с процессором, начиная от P-133 МГц, и работают под управлением Windows 95/98/2000 или Windows NT. Для графического редактора схем ACCEL Schematic требуется ОЗУ 32 Мб, графического редактора печатных плат ACCEL P-CAD PCB - 32 Мб, программы SPECCTRA - 64 Мб.

1.3.7. Сравнительный анализ рассмотренных систем

В настоящее время на рынке программных продуктов имеется довольно много программных средств, позволяющих моделировать работу цифровых устройств. Их основные различия заключаются:

1. В функциональных возможностях;
2. В «дружественности» интерфейса с пользователем;
3. В быстродействии программной модели;
4. В требованиях к аппаратному и программному обеспечению;
5. В адаптируемости к учебному процессу;
6. В возможности использования в телекоммуникационной среде.

С точки зрения настоящей работы больший интерес представляют пункты 1, 2, 5, 6. Именно их мы выберем в качестве основного критерия сравнения существующих систем поддержки изучения цифровой схемотехники. Что касается остальных, то быстродействие, как правило, величина обратно пропорциональная функциональности. Требования к аппаратному и программному обеспечению зависят от выбранного инструментария для реализации модели, а он в свою очередь чаще от конкретной ориентации на область применения программной модели (для образовательных целей в учебных классах, для производственных целей и т.д.).

В предыдущих параграфах были рассмотрены промышленные системы схемотехнического проектирования разных производителей (Micro-Logic, P-CAD), а также исследовательские разработки в этой области других ВУЗов (учебная система «Последовательностные цифровые устройства», учебная система «Проектирование цифровых устройств на вентильных матрицах», HLCCAD). Все они являются либо DOS либо WINDOWS приложениями.

Функциональные возможности рассмотренных систем различны. Промышленные схемотехнические САПР обладают широким спектром возможностей, но практически не пригодны для использования в учебном процессе, так как они ориентированы на специалистов в области конструирования цифровых схем каковыми студенты не являются. В отличии от промышленных САПР, научно-исследовательские системы поддержки изучения цифровой схемотехники изначально ориентированы на учебный процесс, что свидетельствует в их пользу. Но количество таких систем еще очень мало и они разрабатываются в рамках поддержки изучения каких-либо отдельных дисциплин, что накладывает отпечаток на их функциональные возможности.

Что касается оценки «дружественности» интерфейса рассмотренных реализаций, то это субъективный параметр, который значительно изменяется в зависимости от круга лиц, на которых рассчитана система (например, обучаемые и инженеры-специалисты в данной области). Единственное, что можно отметить – это то что промышленные САПР являются в основном зарубежными разработками и имеют англоязычный интерфейс, что так же не способствует их использованию в учебном процессе.

И последнее, ни одна из рассмотренных систем не адаптирована для использования в телекоммуникационной среде.

1.4. Выводы по главе

1. Проблемы построения программно-логических моделей систем поддержки обучения (выбор структуры, алгоритмизация процессов и т.д.) достаточно хорошо освещены в научной и технической литературе. Однако, не удалось обнаружить более-менее серьезных источников информации о построении систем поддержки изучения цифровой схемотехники и схемотехнического проектирования;
2. Анализ методологии построения систем поддержки обучения и алгоритмизации формальных представлений, структуры вычислительной машины и возможностей позволяет сделать вывод о потенциальной разрешимости проблемы;
3. Использование в разработке Интернет/Интранет технологий позволит создать систему обладающую

целым рядом преимуществ: высококачественное представление информации; распределенная, многопользовательская, открытая архитектура; совместимость с основными программно-аппаратными платформами; интеграция с Интернет-ресурсами и возможность использования в процессе дистанционного образования; низкая стоимость реализации всей системы в целом; отсутствие необходимости установки системы на каждое рабочее место; простота организации процессов разработки и развития системы несколькими разработчиками; централизованное обновление версий продукта и постоянная доступность последних версий;

4. Для построения функциональной модели будем использовать формальные представления, рассмотренные в п. 1.1;
5. Применение традиционных схемотехнических стендов для поддержки изучения схемотехнического проектирования малоэффективно в силу их ограниченных функциональных возможностей;
6. Существующие компьютерные автоматизированные системы поддержки изучения схемотехнического проектирования нас не удовлетворяют, потому что большинство из них это промышленные схемотехнические САПР, требующие больших накладных расходов времени студентов на их освоение. Эти САПР не ориентированы на учебный процесс и рассчитаны на профессионального инженера, каковым студент не является.

Глава 2. Исследование подходов к построению учебных схемотехнических САПР и разработка структуры основных компонетов подсистемы поддержки изучения схемотехнического проектирования в ТССМ

2.1. Определение общего подхода к построению подсистемы поддержки изучения схемотехнического проектирования

Результатом любого процесса обучения должны являться знания предметной области, накапливаемые обучаемым [25, 26, 34, 35]. Известно, что знания складываются из следующих составляющих:

$$\textit{Знания} = \textit{информация} + \textit{правила ее использования} + \textit{навыки использования} + \textit{извлечение из информации практического результата} \quad (2.1.1)$$

Взяв за основу данную формулу и анализируя методы построения систем поддержки обучения, архитектуры и функциональных возможностей существующих систем поддержки изучения проектирования цифровых устройств, интернет/интранет технологий разработки программного обеспечения возможно сформулировать общий подход для разработки подсистемы поддержки изучения цифровой схемотехники в ТССМ. Он заключается в совокупности следующих принципов:

1. При концептуальном проектировании подсистемы поддержки изучения цифровой схемотехники целесообразно использовать концепцию построения интеллектуальных гипермедийных систем, сформулированную в п. 1.1.2. Данный принцип позволяет наиболее эффективно реализовать предоставление различной учебно-методической информации обучаемому посредством языка гипертекстовой разметки, дополненному встроенными мультимедия компонентами, такими как статическая и анимированная графическая информация. Это могут быть лекционные курсы по схемотехнике, теоретической информатике и др., включающие в себя описание методов построения различных цифровых устройств, методов их функционирования, изображения типовых схем и т.д. Данный принцип есть реализация первой

составляющей формулы 2.1.1, т.е. качественное предоставление информации.

2. Для реализации второй составляющей формулы 2.1.1 в процессе обучения проводятся практические занятия или лабораторные работы, в ходе которых обучаемые постоянно обращаются к пройденному материалу для его закрепления. Применительно к цифровой схемотехнике это могут быть практические занятия, на которых рассматриваются типовые принципы построения моделей различных устройств и их функционирование. Необходимо учесть такую работу и реализовать средства поддержки ознакомления с типовыми устройствами (логика, триггеры, регистры, счетчики и т.д.) в ТССМ.
3. Обучаемые должны иметь возможность применять полученные навыки при решении конкретных задач проектирования различных цифровых устройств. Эта возможность может быть предоставлена путем реализации средств построения моделей цифровых устройств и моделирования их функционирования. Важную роль при построении моделей цифровых устройств играют знания основных законов алгебры логики и минимизация функций алгебры логики (ФАЛ). Целесообразно использовать для поддержки изучения средство минимизации ФАЛ. Данное средство может не только избавить от рутинного труда, но и для некоторых видов учебной работы стать «решателем», проверяющим результат ручной минимизации. Также можно заложить и пошаговый контроль процессов минимизации, когда промежуточные результаты работы обучаемого контролируются системой. Реализация описанных возможностей есть третья составляющая формулы 2.1.1.
4. Под последней составляющей формулы 2.1.1 подразумевается проведение контрольных или экзаменационных работ, когда обучаемому предлагаются различные не типовые задачи, а он должен их правильно решить. При схемотехническом проектировании такими задачами являются задания на разработку каких-либо цифровых устройств с заданными параметрами их функционирования. Поэтому

необходимо разработать средства тестирования цифровых схем (контроля их качества). Данные средства должны уметь оценить степень соответствия разработанной обучаемым цифровой схемы (комбинационной, последовательностной и т.д.) заданию.

5. Система поддержки изучения должна строится с ориентацией на достаточно активный процесс продолжающейся разработки, в которой уточняется и оптимизируется ее модель и функциональные возможности.
6. В целях повышения эффективности работы пользователя с подсистемой должны обеспечиваться:
   - простой и интуитивно понятный пользовательский интерфейс, возможность его гибкой настройки для адаптации к кругу решаемых задач;
   - встроенная помощь по самой подсистеме и возможность интеграции с существующими информационными ресурсами.
7. Функционирование в телекоммуникационной среде предполагается строить на основе использования Java-технологии, как наиболее эффективной и перспективной для создания сред поддерживающих процесс обучения.

Для обеспечения возможности дальнейшего развития системы необходимо достаточно полно и ясно документировать программный код, придерживаться определенного стиля программирования, а также заранее предусмотреть механизмы централизованного обновления версий системы (например, через WWW сервер).

Для организации информационной поддержки при работе с системой разумно активно использовать интеграцию посредством WWW технологии с уже существующими электронными материалами (справочниками по цифровой схемотехнике, обучающим курсам по автоматизированному проектированию, заданиями на лабораторные и курсовые работы и др.). Это дает возможность обновления и развития справочной, методической и другой информации независимо от ТССМ. Для создания помощи по ТССМ и подсистеме поддержки

изучения проектирования цифровых устройств следует использовать гипертекстовый формат HTML.

2.2. Общая функциональная структура ТССМ и подсистемы поддержки изучения схемотехнического проектирования

На рис. 6. Представлена общая функциональная структура разрабатываемой ТССМ [28]. Прежде чем обратиться непосредственно к структуре средств поддержки изучения проектирования цифровых устройств рассмотрим основные проектные компоненты ТССМ, их назначение и предполагаемые функциональные возможности:

- **Диалоговая оболочка** – должна служить интегрированной средой пользователя ТССМ. Диалоговая оболочка должна реализовывать «дружественный» графический интерфейс взаимодействия с пользователем. Посредством нее необходимо организовывать ввод/вывод данных и межмодульные транзакции.
- **Редактор схем** – должен быть WYSIWYG (What You see is what You get) средством для создания цифровых схем из набора простых элементов (логика, триггеры, мультиплексоры, шифраторы и т.д.) и готовых схемных решений или их редактирования. По функциональности редактор должен быть аналогичен редакторам систем подобного рода, хотя его функциональность может быть несколько меньше. Основные функции, предполагаемые для реализации: вставка и удаление частей схемы или отдельных элементов; соединение контактов элементов проводниками; увеличение/уменьшение обзора (лупа); изменение параметров элементов, являющихся частью схемы; ввод/вывод готовых схемотехнических решений в файлы и базы данных; также необходимо учесть возможность создания пользователем текстовых комментариев к схеме.
- **Буфер обмена** – должен быть непосредственно связан с редактором схем и реализовывать функции копирования участков схемы и их дальнейшую вставку и размножение. В качестве буфера обмена не может использоваться системный буфер обмена операционной системы

(по технической невозможности использования такового из-за ограничений, накладываемых на Java-апплеты), однако он должен позволять обмениваться кусками схем не только внутри одного окна редактора ТССМ, но и между разными окнами.

- **Эмулятор работы схем** – должен выполнять функции моделирования работы цифровых схем с учетом заданных начальных условий. Необходимо учесть, что результаты моделирования могут использоваться различными модулями системы. Во-первых, для проверки работоспособности создаваемых схем. В этом случае они просто визуализируются в виде временных диаграмм. Во-вторых, функции эмулятора должны использоваться модулем тестирования схем на соответствие заданию. В этом случае модуль тестирования должен анализировать правильность работы схемы, основываясь на результатах моделирования.
- **Модуль тестирования** – должен осуществлять выдачу тестового задания пользователю (из файла или базы данных) на проектирование какого-либо цифрового устройства и выполнять тестирование создаваемой схемы на соответствие выданному заданию.
- **Модуль минимизации ФАЛ** – должен выполнять минимизацию функций алгебры логики, что необходимо при проектировании большинства цифровых схем (например, комбинационных). Данное средство позволит сократить время на разработку некоторых схем, что в свою очередь позволит распределить его на другие задачи.
- **СУБД** – необходимо использовать для накопления готовых схемотехнических решений, которые могут быть затребованы пользователем по запросу и для хранения тестовых заданий пользователям, которые также могут выбираться пользователем по запросу.

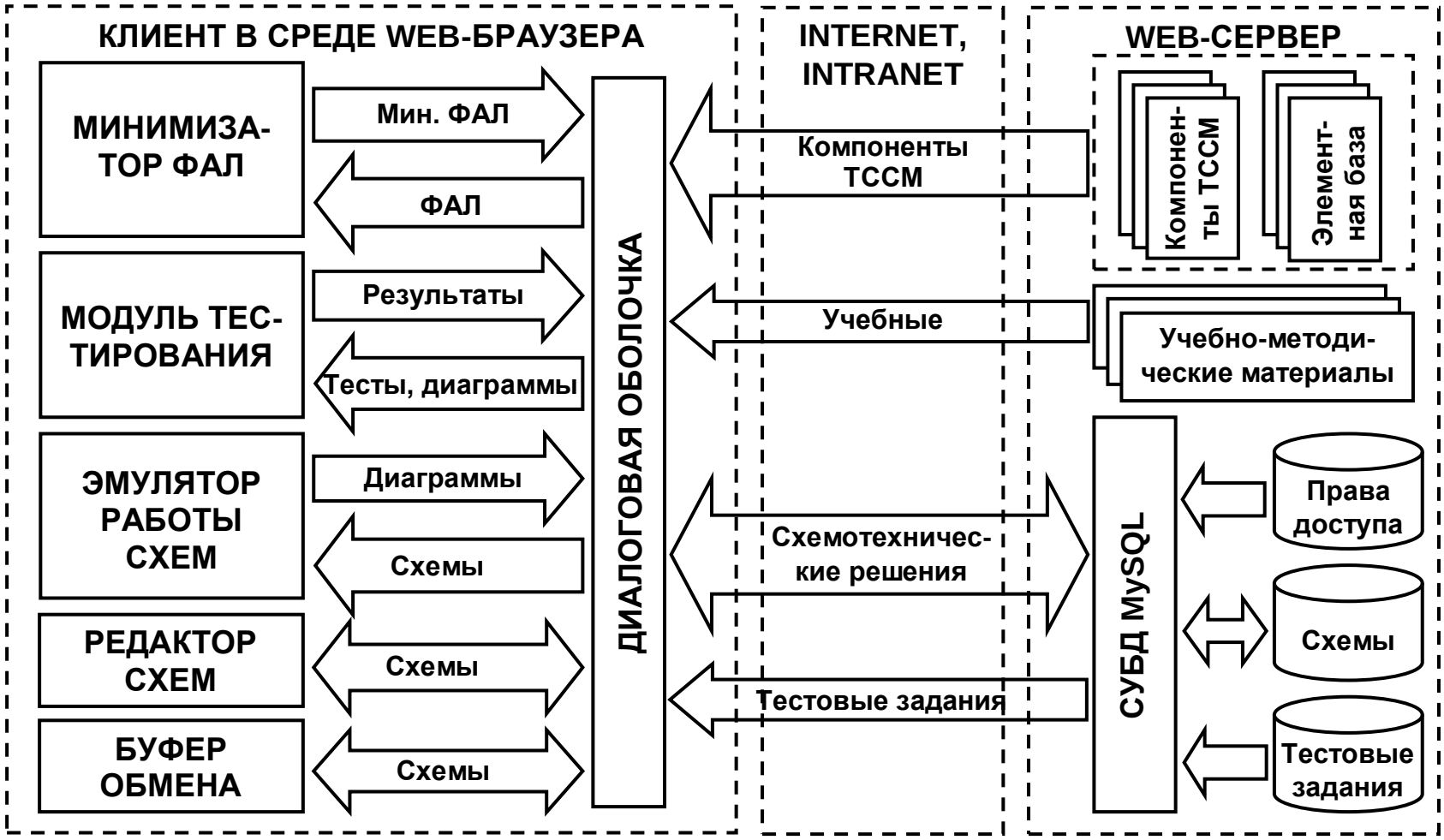


Рис. 6. Функциональная структура ТССМ

- **Подсистема доступа к учебно-методическим материалам** – должна реализовывать удобный классифицированный механизм доступа к уже имеющимся в сети учебно-методическим материалам и осуществлять среди них поиск нужной информации.

Из рассмотренных выше модулей ТССМ к средствам поддержки изучения схемотехнического проектирования можно условно отнести такие компоненты, как модуль минимизации ФАЛ, модуль тестирования и подсистему доступа к учебно-методическим материалам, расположенным в сети Интернет/Интранет. В данной работе в следующих параграфах подробно рассматривается разработка и реализация модуля минимизации ФАЛ и модуля тестирования цифровых схем. Подсистема доступа к учебно-методическим материалам в рамках данной работы не рассматривается.

2.3. Разработка модуля минимизации ФАЛ

2.3.1. Способы задания ФАЛ

Различают несколько способов задания ФАЛ [46, 49], основными из которых являются: табличный, аналитический, координатный, графический, цифровой. При табличном способе ФАЛ задается таблицей истинности, в которой указывается, какое из двух возможных значений «0» или «1» принимает функция на каждом наборе аргументов.

Номер набора	a	b	y1
0	0	0	0
1	0	1	0
2	1	0	1
3	1	1	1

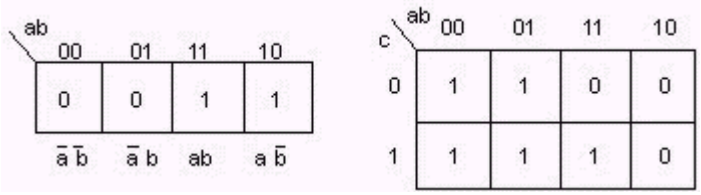
Номер набора	a	B	c	y2
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	1

Аналитический способ предусматривает задание функции в виде формализированного выражения, составленного с использованием математического аппарата АЛ, например:

$$y_1 = \bar{a}b\bar{c} \vee a\bar{b}; \qquad y_2 = (a\bar{c} \vee b)\cdot(b \vee \bar{a})$$

Координатный способ предусматривает задание ФАЛ в виде координатных карт состояний, называемых картами

Карно. При наличии n переменных карты Карно состоят из 2^n полей и представляют собой прямоугольные таблицы, на пересечении строк и столбцов которых записывают значение функции при соответствующем наборе аргументов. При составлении карты необходимо, чтобы наборы аргументов в соседних полях (клетках) отличались только значением одной переменной.



Графический метод предусматривает задание ФАЛ в виде единичного n–мерного куба, вершинам которого соответствуют различные наборы значений аргументов. В каждой вершине указываются значения принимаемые ФАЛ на данном наборе.

При цифровом способе ФАЛ записывается в виде совокупности наборов аргументов, на которых она принимает истинное значение.

y1={2,3}; y2={0-3,7};

2.3.2. Формы представления ФАЛ

Любая функция может быть представлена в виде дизъюнкции (суммы) элементарных произведений или конъюнкции (произведения) элементарных дизъюнкций [47, 49], например

$$f_1 = \bar{a}b \vee \bar{b}c \vee \bar{a}c - ДНФ; \qquad f_2 = (a \vee \bar{b} \vee c)\cdot(\bar{a} \vee c) - КНФ$$

В первом случае функция считается заданной в дизъюнктивной нормальной форме ДНФ, во втором случае – в конъюнктивной нормальной форме КНФ. ДНФ (КНФ) называются совершенными, если все входящие в состав функции элементы произведения (дизъюнкции) в прямом или инверсном виде содержат все переменные алфавита данной функции. Любая ФАЛ имеет только одну ДСНФ и КСНФ. Для получения ДСНФ из таблицы истинности или карты Карно данной функции выписываются все элементарные произведения, соответствующие наборам переменных, на которых ФАЛ принимает единичное значение. При получении КСНФ выписываются все элементарные дизъюнкции, соответствующие наборам переменных, на которых функция превращается в 0, причем каждая из входящих в элементарные дизъюнкции

переменных, инвертируется. Ниже в качестве примера приведена запись ДСНФ и КСНФ функций y_1 и y_2:

$$y_1 = a\bar{b} \vee ab \qquad (ДСНФ)$$
$$y_1 = (a \vee b)\cdot(a \vee \bar{b}) \qquad (КСНФ)$$
$$y_2 = \bar{a}\cdot\bar{b}\cdot\bar{c} \vee \bar{a}\cdot\bar{b}\cdot c \vee \bar{a}\cdot b\cdot\bar{c} \vee \bar{a}\cdot b\cdot c \vee a\cdot b\cdot c \qquad (ДСНФ)$$
$$y_2 = (\bar{a} \vee b \vee c)\cdot(\bar{a} \vee b \vee \bar{c})\cdot(\bar{a} \vee \bar{b} \vee c) \qquad (КСНФ)$$

2.3.3. Основные законы и тождества алгебры логики

Для преобразования функции в алгебре логики используется ряд законов и тождеств, основные из которых приведены ниже.

Коммутативные (переместительные) законы:

$$a \vee b = b \vee a; \qquad a\cdot b = b\cdot a;$$

Ассоциативные (сочетательные) законы:

$$(a \vee b) \vee c = a \vee (b \vee c); \qquad (a\cdot b)\cdot c = a\cdot(b\cdot c);$$

Дистрибутивные (распределительные) законы:

$$a(b \vee c) = ab \vee bc; \qquad a \vee bc = (a \vee b)\cdot(a \vee c);$$

Законы повторения:

$$a \vee a \vee \ldots \vee a = a; \qquad a\cdot a\cdot\ldots\cdot a = a;$$

Законы инверсии (двойственности):

$$\overline{a \vee b} = \bar{a}\cdot\bar{b}; \qquad \overline{a\cdot b} = \bar{a} \vee \bar{b};$$

Законы отрицания:

$$a \vee \bar{a} = 1; \qquad a\cdot\bar{a} = 0;$$

Закон двойного отрицания:

$$\bar{\bar{a}} = a$$

Закон поглощения:

$$a \vee ab = a; \qquad a\cdot(a \vee b) = a$$

Закон склеивания:

$$ab \vee a\bar{b} = a$$

Правила операций с константами:

$$\bar{0} = 1; \qquad \bar{1} = 0; \qquad a\cdot 0 = 0; \qquad a \vee 0 = a; \qquad a \vee 1 = 1;$$

Дополнительные тождества:

$$a \vee \bar{a}b = a \vee b; \qquad a(\bar{a} \vee b) = ab.$$

2.3.4. Минимизация ФАЛ методом Квайна-Мак-Класски

Данный метод основывается на задании входящих в ДСНФ функции элементарных произведений в виде двоичных чисел, называемых номерами соответствующих наборов. Кроме номера каждому произведению присваивается определенный индекс, под которым

понимается число единиц в двоичном представлении данного набора. Например:

набор $\bar{a}b\bar{c}$; *номер* **010(2)**; *индекс* **1**;

набор $ab\bar{c}$; *номер* **110(6)**; *индекс* **2**.

В результате реализации данного метода ФАЛ разлагается на простые импликанты. Под простой импликантой функции понимается всякое элементарное произведение, принимающее единичное значение на всех наборах аргументов, что и исходная ФАЛ, при исключении из которого хотя бы одного аргумента уже не будет выполнятся данное условие.

Алгоритм Квайна-Мак-Класски формулируется следующим образом: для того, чтобы два числа m и n являлись номерами двух склеивающихся между собой наборов, необходимо и достаточно, чтобы индексы данных чисел отличались на единицу, сами числа отличались на степень числа два и число с большим индексом было больше числа с меньшим индексом.

Реализацию алгоритма рассмотрим на примере минимизации ФАЛ

$$f = \bar{a}\cdot\bar{b}\cdot\bar{c}\cdot d \vee \bar{a}\cdot b\cdot\bar{c}\cdot d \vee a\cdot\bar{b}\cdot\bar{c}\cdot d \vee \bar{a}\cdot b\cdot c\cdot d \vee a\cdot\bar{b}\cdot c\cdot d \vee \bar{a}\cdot\bar{b}\cdot c\cdot d$$

На первом этапе минимизации определяем номера и индексы каждого набора, записывая ФАЛ в виде

$$f = 0001 \vee 0101 \vee 1001 \vee 0111 \vee 1011 \vee 0011$$
$$1(I) \quad 5(II) \quad 9(II) \quad 7(III) \quad 11(III) \quad 3(II)$$

Группируем наборы располагая их в порядке возрастания индексов.

Индексы	номера	Результат склеивания		
I	0001(1)	00-1 0-01 -001	0- -1 -0-1 0- -1 -0-1	
II	0011(3) 0101(5) 1001(9)	0-11 -011		
III	0111(7) 1011(11)	01-1 10-1		

На следующем этапе производим склеивание различных наборов, руководствуясь приведенной выше формулировкой алгоритма. Подлежащие склеиванию пары чисел указаны стрелками. При склеивании не совпадающие в числах разряды отмечаются прочерками. Например склеивание чисел 0001 и 0011 дает число 00-1. Результат склеивания выписывается в следующий

столбец таблицы, так же разделяемый на строки с индексами, отличающимися на единицу. После склеивания всех групп первого столбца таблицы переходят ко второму столбцу, вписывая результат склеивания в третий столбец. При объединении наборов второго и последующих столбцов таблицы возможно склевать только числа содержащие прочерки в одноименных разрядах. Склеивание продолжается до тех пор, пока образование нового столбца станет невозможным.

По окончании склеивания приступают к построению импликантной таблицы, записывая в нее в качестве простых импликант наборы, содержащиеся в последнем столбце предыдущей таблицы. В качестве простых импликант так же вписываются наборы из других столбцов предыдущей таблицы не принимавшие участия в склеивании. Если импликанта, содержащаяся в I-ой строке таблицы, составляет некоторую часть конституенты I-го столбца на пересечении I-ой строки и I-го столбца ставится символ *. С целью получения минимальной формы ФАЛ из полученной таблицы необходимо выбрать минимальное число строк, чтобы для каждого столбца, среди выбранных строк нашлась хотя бы одна, содержащая в этом столбце символ *.

наборы / импликанты	$\bar{a}\,b\,\bar{c}\,d$	$\bar{a}\,b\,\bar{c}\,d$	$a\,b\,\bar{c}\,d$	$\bar{a}\,b\,c\,d$	$a\,\bar{b}\,c\,d$	$\bar{a}\,\bar{b}\,c\,d$
$\bar{a}\,d$	*	*		*		*
$\bar{b}\,d$	*		*		*	*

Полученная после минимизации ФАЛ записывается в следующем виде:

$$y = \bar{a}d \vee \bar{b}d.$$

2.3.5. Функциональная структура модуля минимизации ФАЛ

При разработке модуля минимизации ФАЛ изначально было принято решение, что модуль должен быть ориентирован на выполнение следующих функций:

1. Прием ФАЛ заданной в СДНФ от пользователя через «дружественный» графический интерфейс. СДНФ была выбрана вследствии того, что это наиболее часто употребляемая форма представления ФАЛ при проектировании комбинационных, последовательностных и др. цифровых схем. В любом случае какую-либо ФАЛ

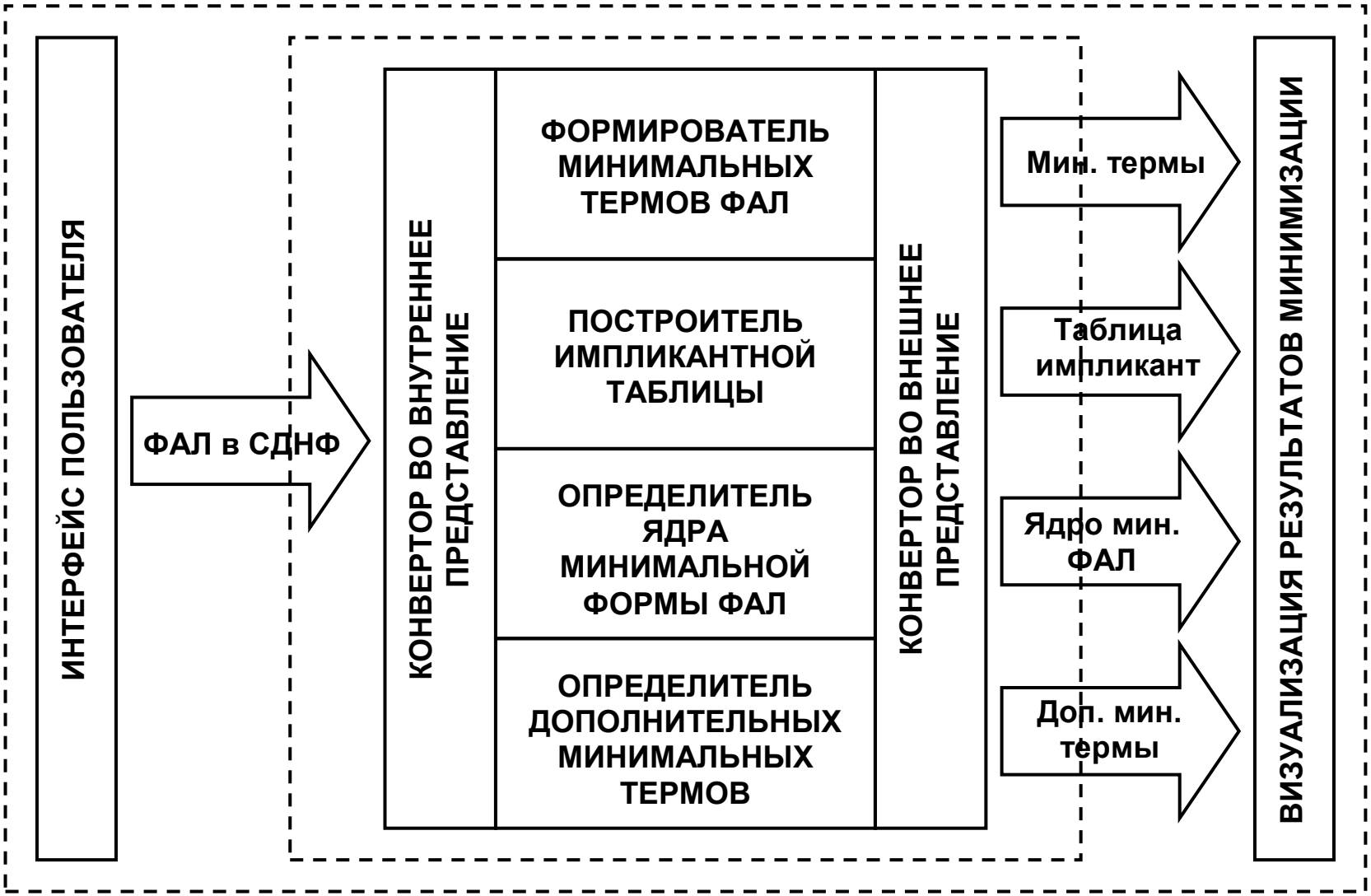


Рис. 7. Функциональная структура модуля минимизации ФАЛ

всегда можно привести к СДНФ, используя свойства алгебры логики (п. 2.3.3.);

2. Минимизация ФАЛ методом Квайна-Мак-Класски. Этот метод выбран, как наиболее четко алгоритмизированный и легко программно реализуемый;
3. Выдача результатов минимизации через «дружественный» графический интерфейс как в виде минимимальной формы ФАЛ, так и в виде отдельных компонентов минимальной ФАЛ (минимальные термы, импликантная таблица, ядро минимальной формы ФАЛ, дополнительные термы);

Учитывая поставленные выше задачи, основные положения алгебры логики, форматы представления и свойства ФАЛ была спроектирована общая функциональная структура модуля минимизации ФАЛ (рис. 7).

2.4. Разработка модуля тестирования цифровых схем на соответствие заданию

2.4.1. Основные подходы к тестированию цифровых схем

Классическая стратегия тестирования цифровых схем основана на формировании тестовых последовательностей, позволяющих обнаруживать заданные множества их неисправностей [36, 50]. Тестовые последовательности обычно задаются генераторами тестовых последовательностей. Для реализации генератора тестовой последовательности желательно использовать простейшие методы, позволяющие избежать сложной процедуры их синтеза. К ним относятся следующие алгоритмы:

- формирование всевозможных тестовых наборов, то есть полного перебора двоичных комбинаций. В результате применения подобного алгоритма генерируются счетчиковые последовательности;
- формирование псевдослучайных тестовых последовательностей;
- формирование случайных тестовых наборов с требуемыми вероятностями единичного и нулевого символов по каждому входу цифровой схемы.

Основным свойством вышеперечисленных алгоритмов является то, что в результате их применения воспроизводятся последовательности очень большой

длины. На выходах проверяемой цифровой схемы формируются ее реакции, имеющие ту же длину. В связи с этим возникают проблемы их запоминания и хранения. Простейшим решением, позволяющим значительно сократить объем хранимой информации об эталонных выходных реакциях, является получение интегральных оценок, имеющих меньшую размерность. Для этого используются алгоритмы сжатия информации. Такие методы тестирования называются компактными. Для того, чтобы применять метод компактного тестирования, необходимо рационально выбирать алгоритм формирования тестовых последовательностей, регулировать их длину.

Для диагностики цифровых схем особый интерес представляют сигнатурный анализ, и метод счета единиц, которые находят широкое применение при реализации компактного тестирования. В основе построения этих методов лежат алгоритмы сжатия информации.

При сигнатурном анализе [39, 40, 51] для сжатия длинных двоичных последовательностей и получения кодов сигнатур используется сигнатурный анализатор, основу которого составляет сдвиговой регистр или его программная реализация с внутренними обратными связями, замыкаемыми через сумматор по модулю 2, на вход которого также поступает последовательность бит, снимаемая с контролируемой точки.

Сигнатурный анализ основывается на следующем принципе сжатия данных: двоичная последовательность x в виде информационного полинома $G(x)$ поступает с выхода проверяемой схемы на сдвиговый регистр и делится в виде полинома $x^kG(x)$ (где k - количество разрядов сдвигового регистра) на порождающий полином $P(x)$ степени k. Деление на порождающий полином $P(x)$ реализуется с помощью сдвигового регистра с обратными связями. Результатом деления является остаток $R(x)$, получающийся в сдвиговом регистре после приема входной последовательности.

Математически процесс описывается формулой:

$$x^kG(x) = Q(x)\,P(x) \oplus R(x),$$

где $Q(x)$ – частное; $R(x)$ – остаток; $P(x)$ – порождающий полином степени k, а $G(x)$ – информационный полином, соответствующий входной двоичной последовательности x.

При прохождении последовательности x через сдвиговый регистр $R(x)$ изменяется до тех пор, пока не закончится вся последовательность x. Конечное выражение $R(x)$ является сигнатурой.

И в итоге алгоритм работы заключается в подаче на вход цифровой схемы входной последовательности, считывании выходной последовательности с ее выхода и сложении по модулю 2 с контрольной последовательностью.

В случае, если результат сложения не равен логическому «0» (выходная и контрольная последовательности не равны друг другу), то цифровая схема функционирует неверно. Если же в результате сложения всех входных и контрольных последовательностей по модулю 2 получили логический «0» - цифровая схема работоспособна.

Построение различных сложных цифровых схем требует повышенного внимания к компактным методам тестирования для каждого конкретного применения. Поэтому возникает необходимость в оценке эффективности того или иного метода компактного тестирования. Доминирующей мерой эффективности является его достоверность, которая определяется вероятностью необнаружения ошибок определенной кратности в диагностируемой схеме. В качестве характеристики, позволяющей оценить метод компактного тестирования целесообразно использовать распределение вероятностей необнаружения ошибки в зависимости от ее кратности, которое определяется произведением вероятностей возникновения ошибки и необнаружения возникшей ошибки. Значение вероятности возникновения ошибки определяется видом проверяемой цифровой схемы, множеством возможных ее неисправностей, а также видом тестовых последовательностей. Вероятность необнаружения возникшей ошибки характеризуется только методом компактного тестирования. Анализ этого вида распределения позволяет принимать решение о целесообразности применения того или иного метода компактного тестирования.

2.4.2. Метод тестирования цифровых схем в ТССМ и структура тестовых заданий

Рассмотренный в предыдущем параграфе метод тестирования цифровых устройств на основе сигнатурного анализа хорошо себя зарекомендовал при оценке работоспособности реальных комбинационных схем и интегральных микросхем. Однако, использование этого метода без модификации для тестирования различных моделей цифровых устройств в ТССМ, особенно последовательностных, не приемлемо в силу ряда причин:

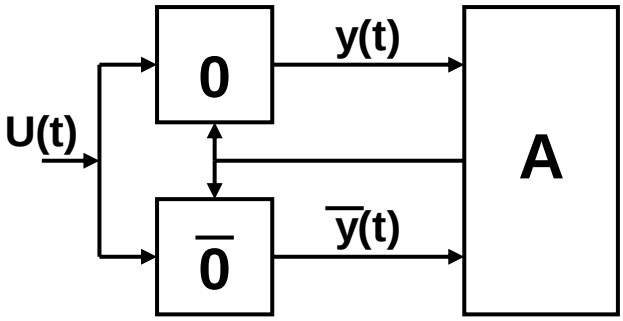
1. Данный метод наиболее ориентирован на тестирование устройств в статическом режиме работы, т.е. когда задержки распространения сигнала не учитываются;
2. При фиксированном размере вычисляемой сигнатуры вероятность необнаружения ошибки растет с увеличением времени моделирования;
3. При несовпадении контрольной сигнатуры с полученной можно констатировать факт наличия ошибки в работе схемы, но определить промежуток времени, в который она произошла, затруднительно.

Исходя из вышеперечисленного целесообразно отказаться от фиксированного размера сигнатуры и строить ее по другому принципу.

Управляющие последовательности, поступающие на входы схемы и последовательности, получаемые в результате моделирования с ее выходов, разбиваются на интервалы. На интервале необходимо так прикладывать управляющие воздействия, чтобы состояние на каждом из выходов схемы в этом интервале менялось не более одного раза. В таком случае контрольная сигнатура на каком-либо выходе схемы будет строиться на основании изменений состояния на каждом интервале полученной на этом выходе последовательности значений. Таким образом i-ая позиция в сигнатуре будет принимать следующие значения:

- «0», если на i-ом интервале состояние на выходе меняется из «0» в «1» или не меняется вовсе;
- «1», если на i-ом интервале состояние на выходе меняется из «1» в «0» или не меняется вовсе;

Необходимо так выбирать длину интервала, чтобы учесть влияние задержек на элементах на распространение сигнала. Длина интервала должна быть больше, чем максимально возможная суммарная задержка прохождения сигнала от входов схемы к ее выходам. Полученные сигнатуры сравниваются с эталонными (п. 2.4.1). Но в результате сравнения сигнатур,



U(t) – управляющие воздействия на t-ом интервале;
y(t) – эталонная метка;
$\overline{y}$(t) – полученная метка;
A – анализатор.

Рис. 8. Схема сравнения меток.

полученных описанным способом можно определить не только возникновение ошибки, но и временной интервал, на котором она произошла. Модель процесса сравнения меток имеет вид, представленный на рис. 8.

Исходя из вышесказанного, структура теста должна включать в себя информацию об управляющих воздействиях и эталонных сигнатурах, полученных на

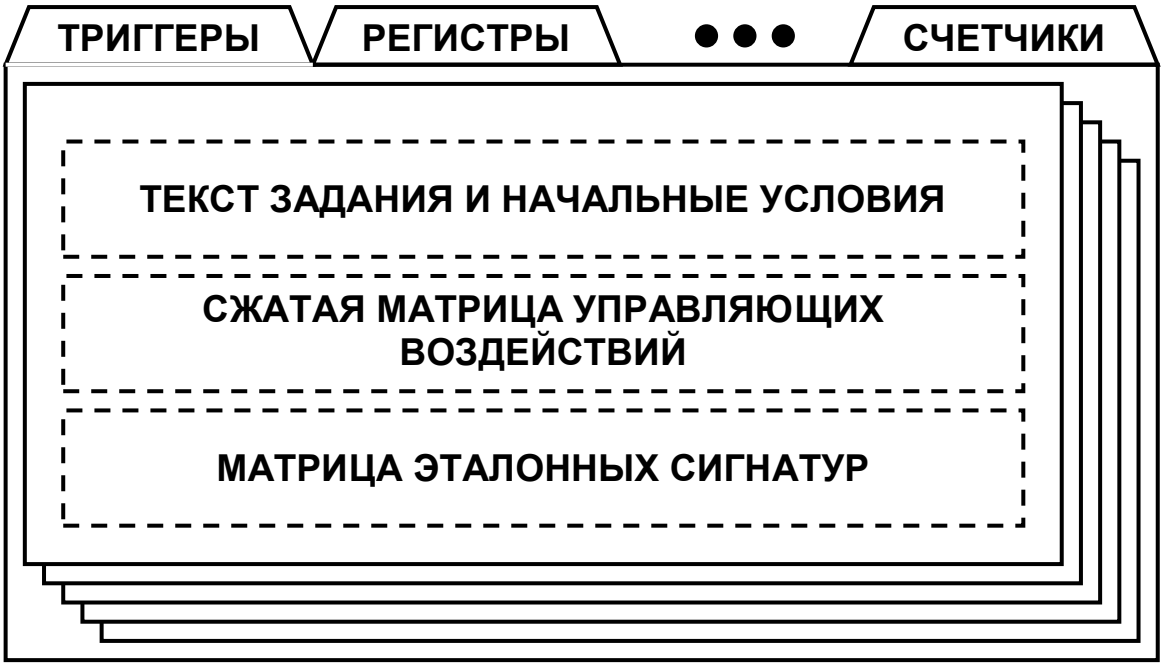


Рис. 9. Структура тестовых заданий

выходах правильно работающей схемы. Также необходимо присутствие информации о типе схемы и ее параметрах

(текст задания), необходимых обучаемому для правильного проектирования схемы. Предполагаемая структура теста представлена на рис. 9.

2.4.3. Функциональная структура модуля тестирования

Модуль тестирования (контроля качества) цифровых схем должен быть ориентирован на выполнение следующих задач:

1. Доступ пользователя к тестовым заданиям, расположенным в сети, в файле, в БД;
2. Визуальное представление вариантов заданий на разработку цифровых схем с возможностью выбора задания из соответствующей категории (счетчики, триггеры, регистры и т.д.);
3. Визуальное представление пользователю подробной текстовой формулировки выбранного задания, включающего в себя формулировку условий, которые должны быть обязательно выполнены при проектировании схемы (присутствие в схеме генераторов и маркеров с определенными символьными метками);
4. Выполнение тестирования (проверки работоспособности) созданной пользователем цифровой схемы на соответствие выбранному заданию;
5. Формирование и визуальное представление пользователю результатов тестирования;
6. Формирование и визуальное представление пользователю советов по изменению проектируемой схемы или параметров входящих в нее элементов в случае не успешного завершения тестирования;
7. Визуализация пользователю диаграмм управляющих воздействий прикладываемых к тестируемой цифровой схеме и предполагаемых эталонных диаграмм ее работы;
8. Визуализация пользователю диаграмм работы цифровой схемы, полученных в результате моделирования;

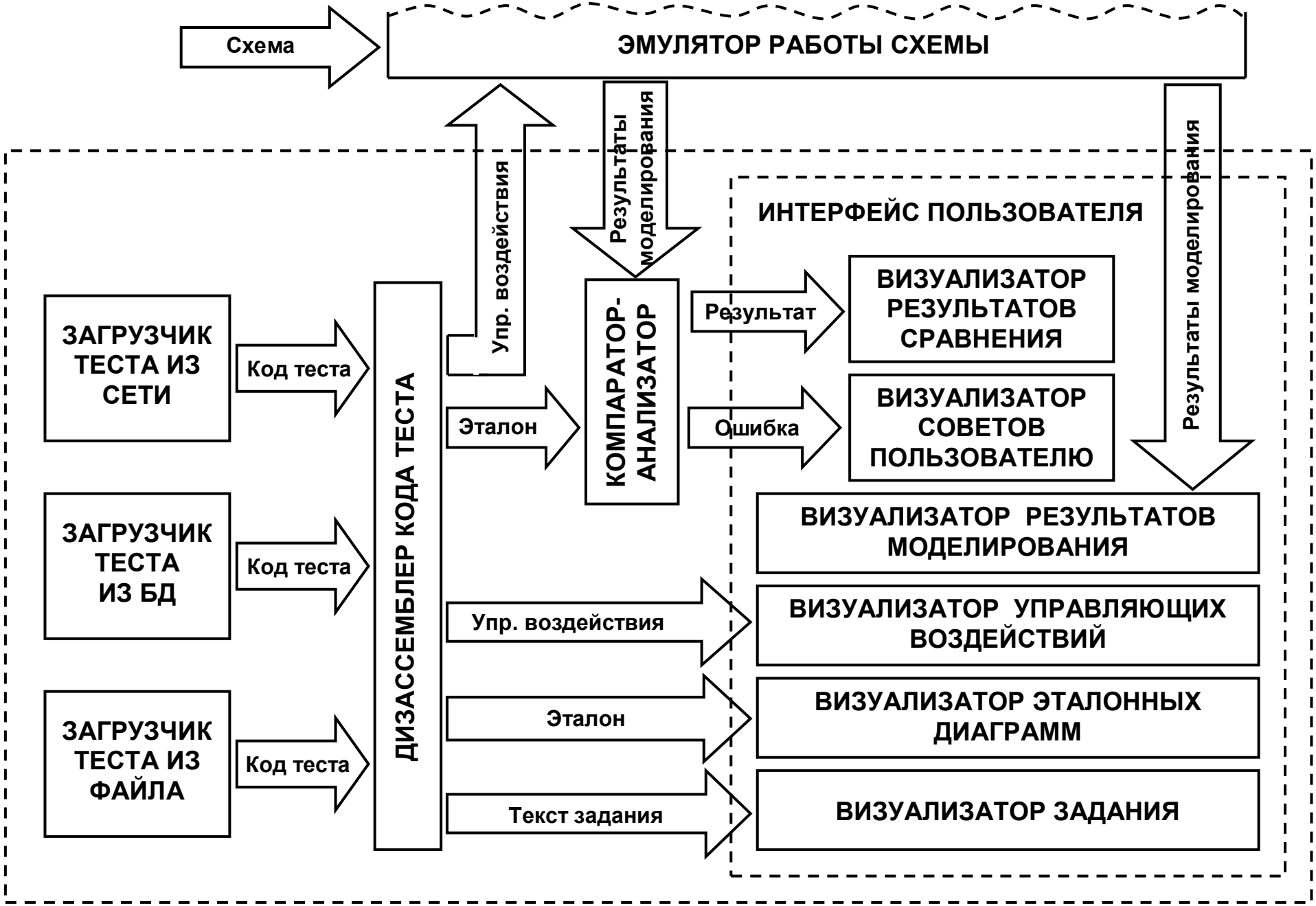


Рис. 10. Функциональная структура модуля тестирования

Учитывая поставленные задачи, структура модуля тестирования (рис. 10) должна в себя включать следующие основные компоненты:

- Загрузчики кода теста, предоставляющие пользователю выбор загрузки теста из сети, из файла, из БД;
- Дизассемблер (разборщик) кода теста (п. 2.4.2), выполняющий анализ текстового кода теста;
- Компаратор-анализатор – блок сравнения диаграмм результатов моделирования работы цифровой схемы с предполагаемыми эталонными диаграммами;
- Компоненты-визуализаторы текста задания, диаграмм управляющих воздействий, диаграмм результатов моделирования, эталонных диаграмм, результатов тестирования, а также советов пользователю в случае неправильной работы созданной им схемы;

2.5. Выводы по главе

1. Сформулирован общий подход для разработки средств поддержки изучения схемотехнического проектирования, ориентированных на использование в телекоммуникационной среде.
2. Определена общая функциональная структура телекоммуникационной системы схемотехнического моделирования и средств поддержки изучения схемотехнического проектирования.
3. Определена функциональная структура модуля минимизации ФАЛ. Модуль был спроектирован только как средство облегчения рутинной работы по минимизации ФАЛ, в него не были включены никакие другие функциональные возможности, рассмотренные в п. 2.1.
4. Рассмотрены подходы к тестированию (контролю качества) работы цифровых схем и устройств и сформирован метод тестирования (контроля качества) цифровых схем в ТССМ. На основе этого построена структура модуля тестирования (контроля качества).
5. Разработана структура учебных контрольных тестовых заданий для проверки работоспособности цифровых схем.

Глава 3. Программные реализации модулей и экспериментальные исследования применимости средств поддержки изучения схемотехнического проектирования в ТССМ

3.1. Программная реализация модуля минимизации ФАЛ

Модуль минимизации ФАЛ представляет собой двухкомпонентную структуру (п. 2.4.3), где один компонент призван решить задачи взаимодействия с пользователем (ввод ФАЛ, редактирование ФАЛ, вывод результатов минимизации), а другой непосредственно реализует функцию минимизации ФАЛ методом Квайна-Мак-Классски, что позволяет использовать его функциональность при решении других задач.

При программной реализации логично построить два класса, реализующих вышеописанные функции.

Первый класс наследуем от класса Dialog и реализуем функции ввода, редактирования ФАЛ и отображения результатов минимизации:

```
package schematics.program;

import java.awt.*;
import java.util.*;
import java.awt.event.*;

public class minimize_dialog extends Dialog implements WindowListener,
ActionListener, FocusListener{

	// Лист термов ФАЛ
	private List termsList = new List();

	// Диалоговые элементы
	private Button minimizeButton = new Button("Минимизировать");
	private Button clearButton = new Button("Очистить");
	private Button addTermButton = new Button("+");
	private Button delTermButton = new Button("-");
	private Button cancelButton = new Button("Отменить");
	private nice_panel MainPanel = new nice_panel("background.jpg");

	// Поля ввода
	private TextArea resultArea = new TextArea();
	private TextField numVarsField = new TextField();
	private TextField termField = new TextField();

	// Количество переменных в термах ФАЛ
	private int numVars = 0;

	public minimize_dialog(Frame parent){
		super(parent, "Минимизация булевой функции", true);
		...
```

```
            Расположение управляющих элементов и полей ввода/вывода
            ...
      }

      private void minimizeButtonClicked(){
            ...
            Создание экземпляра класса, реализующего минимизацию ФАЛ
            ...
            Вывод результатов минимизации
            ...
      }

      private void clearButtonClicked(){
            ...
            Удаление всех термов ФАЛ для последующего ввода новой ФАЛ
            ...
      }

      private void addTermButtonClicked(){
            ...
            Добавление терма в ФАЛ
            ...
      }

      private void delTermButtonClicked(){
            ...
            Удаление терма из ФАЛ
            ...
      }

      public void actionPerformed(ActionEvent e){
            ...
            Обработка событий от управляющих элементов и полей ввода
            ...
      }

      public void windowClosing(WindowEvent e){
            ...
            Закрытие окна диалога
            ...
      }
}
```

Второй класс выполняет только функции по минимизации ФАЛ, поэтому наследовать его от чего бы то ни было нет необходимости. Классу при инициализации передается толко вектор термов минимизируемой ФАЛ:

```
package schematics.program;

import java.util.*;

public class binary_function {
      private Vector terms;    // термы ФАЛ
      private Vector minTerms; // минтермы
      private Vector core;     // ядро минимальной ФАЛ
      private Vector extTerms; // дополнительный минтермы
```

```
private int [][] matrix; // импликантная таблица
int termLength;          // кол-во переменных в терме

public binary_function(Vector t){
      terms = new Vector();
      for(int i = 0; i < t.size(); i++)
      terms.addElement(stringToTermArray((String)t.elementAt(i);
      removeDoubles(terms);
      termLength = makeTermLength();
      minTerms = makeMinTerms();
      matrix = makeMatrix();
      core = makeCore();
      extTerms = makeExtTerms();
}

private int makeTermLength(){
      ...
      Определение количества переменных в термах ФАЛ
      ...
}

private Vector makeMinTerms(){
      Формирование минтермов
      Vector src = terms;
      Vector dst = new Vector();
      while(nextCub(src, dst)){
            src = dst;
            dst = new Vector();
      }
      return dst;
}

public String [] getMinTerms(){
      ...
      Выдача минтермов в виде массива строк
      ...
}

public String [] getCore(){
      ...
      Выдача ядра минимальной ФАЛ
      в виде массива строк
      ...
}

public String [] getExtTerms(){
      ...
      Выдача дополнительных минтермов для полного
      покрытия ФАЛ в виде массива строк
      ...
}

private boolean nextCub(Vector src, Vector dst){
      ...
      Склеивание термов очередного куба
      ...
}

private int [][] makeMatrix(){
      ...
      Построение импликантной таблицы
```

```
	...
}

public Vector makeCore(){
	...
	Определение ядра минимальной ФАЛ
	...
}

public Vector makeExtTerms(){
	...
	Определение дополнительных минтермов для полного
	покрытия ФАЛ
	...
}

private boolean isAdded(Vector dst, int [] array){
	...
	Проверка на присутствие минтерма в векторе минтермов
	...
}

private void removeDoubles(Vector dst){
	...
	Удаление дубликатов термов
	...
}

private boolean equalsIntArrays(int [] src, int [] dst){
	...
	Сравнение двух числовых массивов
	...
}

private boolean termCovered(int [] src, int [] minterm){
	...
	Определение возможности покрытия терма минтермом
	...
}

private int [] stringToTermArray(String str){
	...
	Преобразование текстового представления терма
	во внутренний числовой формат
	...
}

private String termArrayToString(int [] src){
	...
	Преобразование внутреннего числового представления терма
	в текстовый
	...
}

public String [] getFunc(){
	...
	Выдача ФАЛ в виде строки
	...
}

public String getTable(){
```

```
            ...
            Выдача импликантной таблицы в виде строки текста
            ...
        }
}
```

1 – поле ввода очередного терма ФАЛ;
2 – поле ввода длины термов (кол-ва переменных в терме);
3 – кнопка добавления очередного терма ФАЛ;
4 – кнопка удаления выбранного из списка терма ФАЛ;
5 – список термов ФАЛ;
6 – область отображения результатов минимизации ФАЛ;
7 – кнопка очистки всех полей диалогового окна;
8 – кнопка активации процесса минимизации ФАЛ;
9 – кнопка закрытия диалогового окна.

Рис. 11. Графический интерфейс пользователя модуля минимизации ФАЛ

Графический интерфейс пользователя модуля минимизации ФАЛ (рис. 11) представляет собой

диалоговое окно, основными компонентами которого являются элементы ввода и редактирования ФАЛ (1, 2, 3, 4) и текстовая область для отображения протокола результатов минимизации функций (6). Результаты минимизации отображаются в текстовом виде и легко могут быть перенесены через системный буфер обмена в какое-либо другое приложение.

3.2. Программная реализация модуля тестирования цифровых схем

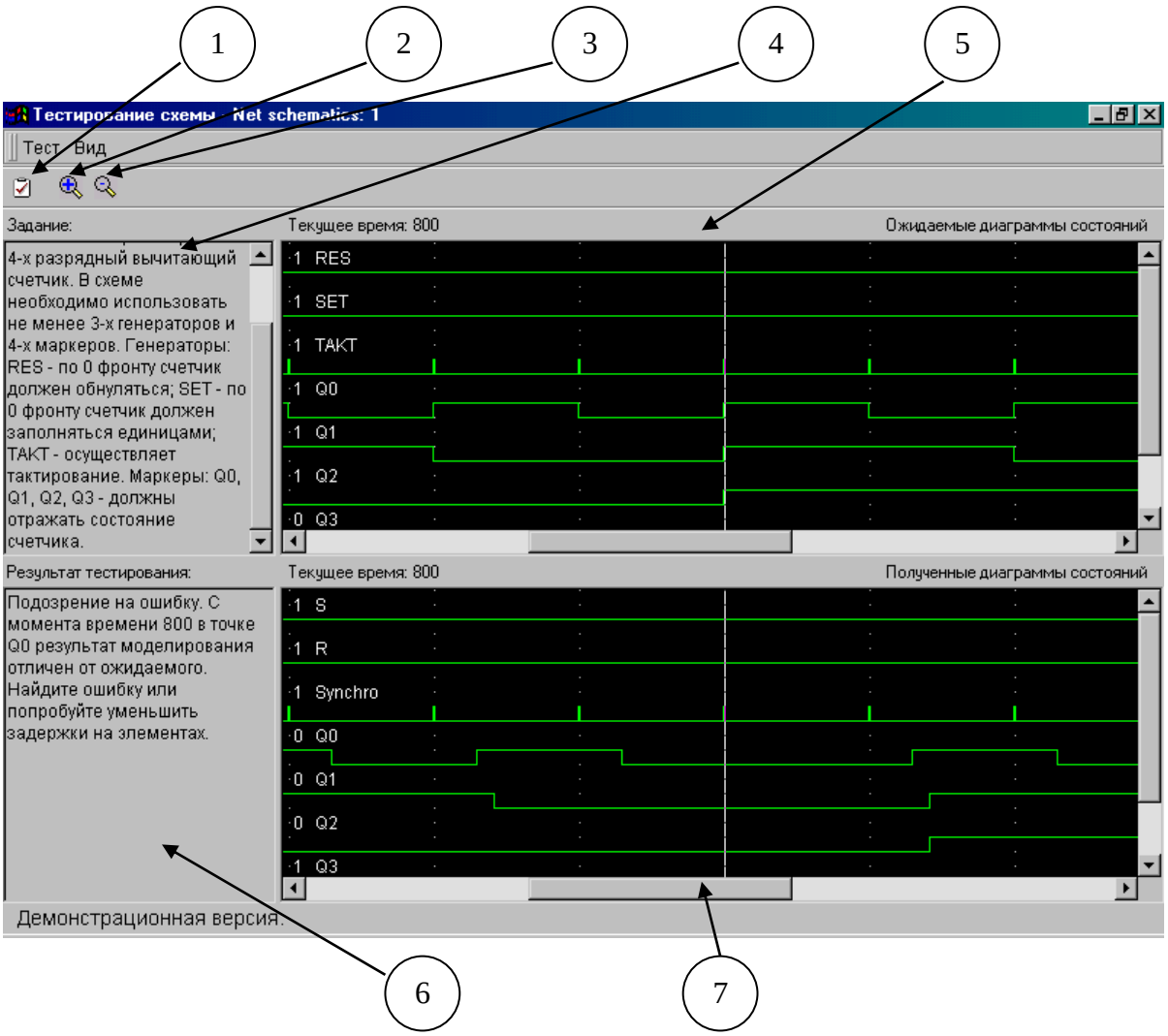
Модуль тестирования цифровых схем (контроля качества) на соответствие заданию реализован на базе класса Window. При разработке класса-наследника учтена сформированная предполагаемая функциональная структура модуля (п. 2.4.3). Элементы интерфейса пользователя, функции работы с тестами (загрузка, визуализация) и тестирование цифровых схем реализованы в одном классе:

```
package schematics.program;

import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.util.*;
import java.io.*;
import schematics.elements.elements.*;

public class test_frame extends Frame{
      button_bar_panel toolBar;      // панель инструментов
      status_bar statusBar;          // панель статуса
      settings curSettings;          // текущие установки
      TextArea taskTextArea;         // область визуализации задания
      TextArea errorTextArea;        // область визуализации результатов
                                     // тестирования и ошибок
      diagrams_panel diagPanel1;     // область визуализации эталонных
                                     // диаграмм
      diagrams_panel diagPanel2;     // область визуализации полученных
                                     // диаграмм
      Hashtable test;                // таблица, содержащая код теста
      Hashtable waiting;             // таблица с эталонными метками
      Vector [] influences;          // массив векторов управляющих
                                     // воздействий

      public test_frame(settings s){
            ...
            Инициализация класса и настройка элементов интерфейса
            ...
      }

      public void initMenuBar(){
            ...
            Инициализация панели меню
            ...
```

```
}

private void initToolbar(){
      ...
      Инициализация панели инструментов
      ...
}

public boolean action(Event e,Object what){
      ...
      Обработка различных команд от элементов управления
      ...
}

private void selectedURLOpen()
      ...
      Инициализация объекта test и загрузка кода теста из сети
      ...
}

public void selectedFileOpen(){
      ...
      Инициализация объекта test и загрузка кода теста из файла
      ...
}

private void disassembleTest(){
      ...
      Разбор кода теста, визуализация задания пользователю,
      инициализация объектов waiting и influences
      ...
}

private void selectedTest(){
      ...
      Моделирование работы цифровой схемы, полученной из редактора.
      Получение меток. Сравнение эталонных меток с полученными.
      Визуализация результатов сравнения и ошибок.
      ...
}

private void selectedZoomout(){
      ...
      Уменьшение масштаба отображения диаграмм
      ...
}

private void selectedZoomin(){
      ...
      Увеличение масштаба отображения диаграмм
      ...
}

private void setStatus(String s){
      ...
      Отображение информации в строке статуса
      ...
}

private void exit(){
      ...
```

```
            Деинициализация и закрытие окна
            ...
        }

        public boolean handleEvent(Event event) {
            ...
            Обработка событий
            ...
        }
}
```

Интерфейс, предоставляемый пользователю модулем тестирования цифровых схем представлен на рис. 12.



1 – кнопка активации процесса тестирования схемы;
2 – кнопка уменьшения масштаба отображения диаграмм;
3 – кнопка увеличения масштаба отображения диаграмм;
4 – область визуализации задания на проектирование схемы;
5 – панель визуализации ожидаемых (эталонных) диаграмм работы схемы;
6 – область визуализации результатов тестирования и советов пользователю;
7 – панель визуализации полученных в результате моделирования диаграмм;

Рис. 12. Графический интерфейс пользователя модуля тестирования цифровых схем

3.3. Разработка учебных цифровых схем и контрольных тестовых заданий

ТССМ ориентирована на поддержку изучения базовых принципов цифровой схемотехники. Обычно в базовые учебные курсы включается изучение основ булевой алгебры, двоичной арифметики, рассмотрение вопросов синтеза и анализа комбинационных и последовательностных схем.

Комбинационные схемы рассматриваются как совокупность следующих групп:

- базовые логические элементы;
- коммутационные узлы (ключи, мультиплексоры, селекторы, в том числе многофункциональные, матричные коммутаторы и др.);
- преобразователи кодов (дешифраторы, шифраторы, приоритетные шифраторы, специализированные преобразователи двоичных кодов);
- постоянные запоминающие устройства (ПЗУ);
- арифметические узлы (четвертьсумматоры, полусумматоры, сумматоры, схемы ускоренного переноса, инкременторы, декременторы, цифровые матричные умножители, цифровые компараторы, арифметико-логические устройства, схемы контроля на четность/нечетность и др.)

Последовательностные схемы (схемы с памятью) представляются совокупностью следующих групп:

- триггерные устройства;
- регистры;
- счетчики;
- пересчетные устройства;
- многостабильные схемы;
- распределители импульсов;
- статические и динамические оперативные запоминающие устройства;

Разработка учебных цифровых схем велась с ориентацией на максимальное покрытие изучаемых групп комбинационных и последовательностных устройств. Однако, были рассмотрены не все группы устройств. С помощью редактора моделей цифровых схем ТССМ были спроектированы модели таких комбинационных схем, как шифраторы, сумматоры, мультиплексоры, схемы формирования нечетности, и последовательностных схем,

таких как триггеры, регистры, счетчики. Всего около 20 различных схем. В следствии большого объема кода схем в приложении 1 представлены 3 модели из всех разработанных устройств. Данные схемы были проверены на работоспособность и, для проведения тестирования модуля контроля качества цифровых схем, на основании их функционирования были созданы учебные тестовые задания. В приложении 2 приведены тестовые задания на разработку цифровых схем, соответствующих учебным схемам из приложения 1. При разработке учебных тестовых заданий использовались подходы, рассмотренные в п. 2.4.1 и 2.4.2.

3.4. Требования обеспечения установки и эксплуатации

Требования к квалификации пользователя. Для продуктивной работы с системой пользователю необходимо:

- Уметь работать в ОС Windows;
- Иметь опыт работы с интернет-браузерами (такими как MS Internet Explorer, Netscape Navigator или Opera) или соответствующим программным обеспечением из комплекта JDK;
- Иметь базовое представление о методах работы в САПР;
- Знать основные понятия логики и цифровой схемотехники;

Требования для установки и эксплуатации системы. Поскольку система разработана как Java-апплет, то снимается необходимость установки на каждое рабочее место. Достаточно разместить систему на WWW-сервере. При таком варианте все рабочие места должны иметь подключение к сети и доступ к WWW-серверу. Кроме того, на них необходимо установить клиентское программное обеспечение для работы с WWW (так называемые интернет-браузеры) поддерживающие выполнение Java-апплетов или же комплект JDK (номера версий см. в пункте «Условия среды эксплуатации»). В случае отсутствия подключения к сети или к WWW-серверу есть возможность запуска системы с локального диска.

Условия среды эксплуатации. Система функционирует на IBM-совместимых ПЭВМ, под управлением ОС Windows 95/98, Windows NT 4.0 и выше. Имеет хорошую

производительность на моделях процессоров, начиная с Intel Pentium-166. Java-апплет функционирует в среде appletviewer из состава JDK 1.1 и выше, а также в интернет-браузерах Internet Explorer (версий 4.01 и выше), Netscape Navigator (версий 4.5 и выше) и Opera (версий 5.0 и выше). В случае размещения системы на WWW-сервере никаких дополнительных настроек в конфигурации сервера делать не нужно, система абсолютно не критична к платформе сервера и его программному обеспечению.

3.5. Измерение и сравнение производительности Java и C++ программ

Цель эксперимента. Целью данного эксперимента является оценка производительности Java Virtual Machine (JVM) разных сред поддерживающих выполнение Java-программ и сравнение этих показателей производительности с результатами, полученными при использовании тестирующей программы реализующей точно такой же алгоритм, но откомпилированной в машинный код архитектуры на которой производится тестирование.

Существует несколько причин, почему Java «исторически» медленнее, чем C (если сравнивать чистую производительность). Java оптимизируется во время выполнения кода, а не во время компиляции, и при этом стандартные библиотеки должны работать одинаково на всех платформах, что требует дополнительного уровня программного обеспечения. Java еще и относительно новая технология - не нужно забывать, что прошло много времени, пока компиляторы C стали хороши (к тому же они и сейчас постоянно улучшаются) - и тоже самое было и с C++. На вершине производительности стоят всегда низкоуровневые программы, использующие маленькие библиотеки или не использующие их вовсе. На пути оптимизации есть несколько трудностей и в Java - любой доступ к массиву проверяется на легальность (с проверкой границ массива), стандартная строковая библиотека использует 16-bit Unicode вместо 8-bit ASCII, что означает, что все строки в два раза больше, все массивы должны быть инициализированы, динамическое распределение памяти работает через главный сборщик мусора, никакие мейнстримы центральных процессоров пока не используются ни в каких оптимизаторах, а так

же имеются трудности с реализацией специфических мультимедийных инструкций. В теории, сборщик мусора и динамическая оптимизация могут быть преимуществом сами по себе, а большая часть трудностей, описанных выше, может быть оптимизирована хорошим оптимизатором – исключая 16-bit представление строк.

Конфигурация оборудования. Для проведения экспериментов использовался персональный компьютер следующей конфигурации: Pentium 120, 32Мб ОЗУ, HDD4.3Gb IDE, ОС Windows 98.

Алгоритмы тестирования [60, 61].

1. "Game of Life" с легкой загрузкой. Это немного продвинутая реализация алгоритма J.H. Conway простой сотовой автоматизации - 2D набор ячеек, которые могут быть или живыми или мертвыми, смерть или жизнь зависит от количества соседних живых или мертвых ячеек. В этой версии все статично, и сильно зависит от доступа к массиву, а так же имеется фиксированное игровое поле с единственным "глидером" (повторяющийся шаблон с 5 живыми ячейками двигающимся по диагонали), поэтому, чем больше становится поле, тем больше алгоритм будет зависеть от простого чтения массива. Это наиболее длинный и сложный код из всех тестов.
2. "Game of Life" с сильной загрузкой. Код идентичен предыдущему, но здесь количество глидеров зависит от размера поля, и их так много, как только возможно. При этом алгоритм уже меньше зависит от доступа к массиву.
3. "Game of Life" с безразмерным полем. Это полностью другая реализация, использующая хеш для имитации разреженной матрицы, который в действительности же имеет максимальный размер 2^{32}x2^{32}. Эта реализация требует динамического распределения памяти и имеет уникальный алгоритм хеширования (для сохранения идентичности алгоритмов в разных языках) но он меньше и проще, чем версия со статической памятью. Для этого теста загрузка - квадратный блок глидеров, сильно сжатых, длина каждой из сторон кратна 5. При этом алгоритм значительно медленнее, чем предыдущие.

4. Серии Fibonacci. Серии Фибоначи это очень простая функция, вызывающая себя рекурсивно дважды на каждой итерации (исключая заключительную итерацию) и вычисляющая время увеличения экспоненциально. Компилятору здесь не очень много чего есть что делать (ну только разве, что он может определить статическую натуру результатов) исключая inline и рекурсивных вызовов. Этот алгоритм представлен как альтернатива алгоритмам с массивами.
5. Простая реализация FFT. В этом тесте происходит множество быстрых преобразований Фурье, но с другой стороны он достаточно прост и содержит всего 4 процедуры с вложенными циклами. Поскольку в тесте используется двойная точность для работы с плавающей точкой, то он еще более отстранен от доступа к массивам.

Тексты программ приведены в приложении 3.

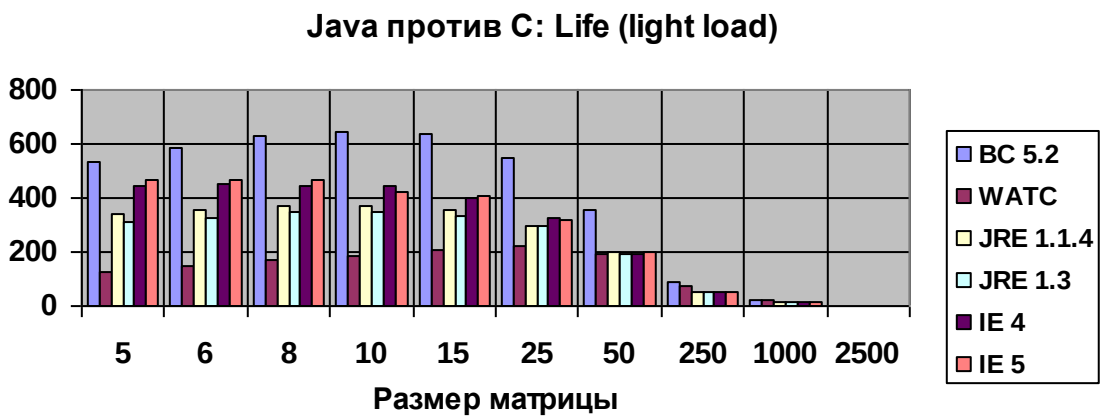
Методы тестирования. Для каждой из программ, было произведено несколько запусков тестов с увеличением сложности или размера, кроме этого, каждый из тестов запускался 5 раз, самые большие и маленькие результаты отбрасывались, а конечным результатом было среднее, между оставшимися тремя. Для каждого теста было осуществлено два быстрых пробных теста для того, что бы определить, сколько нужно совершить циклов, что бы время работы программы было примерно равно 10-ти секундам, это нужно для того что бы уменьшить погрешность при измерении времени. 10-ти секунд так же достаточно, чтобы виртуальная машина полностью заоптимизировалась, так же в это время осуществляется самоподстройка кода, для того, что бы он работал быстрее (оптимизация кода). Известно что в Windows 98 работа таймера (точность), вычислительных потоков и многозадачности, обработка прерываний не удовлетворительна (намного хуже, чем под Solaris, FreeBSD и Linux), но результаты в основном были предсказуемы и стабильны, и зачастую не было больших различий во всех пяти запусках.

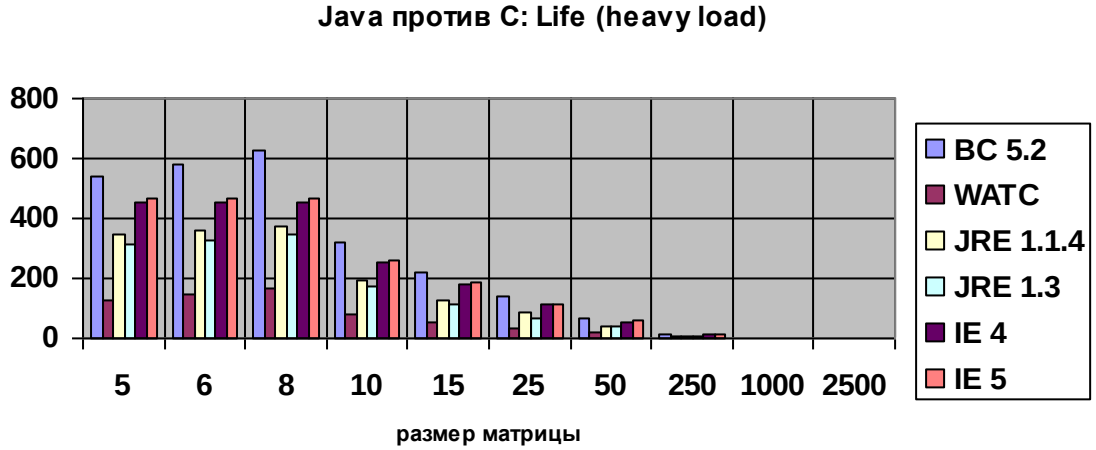
Ключи к графикам.

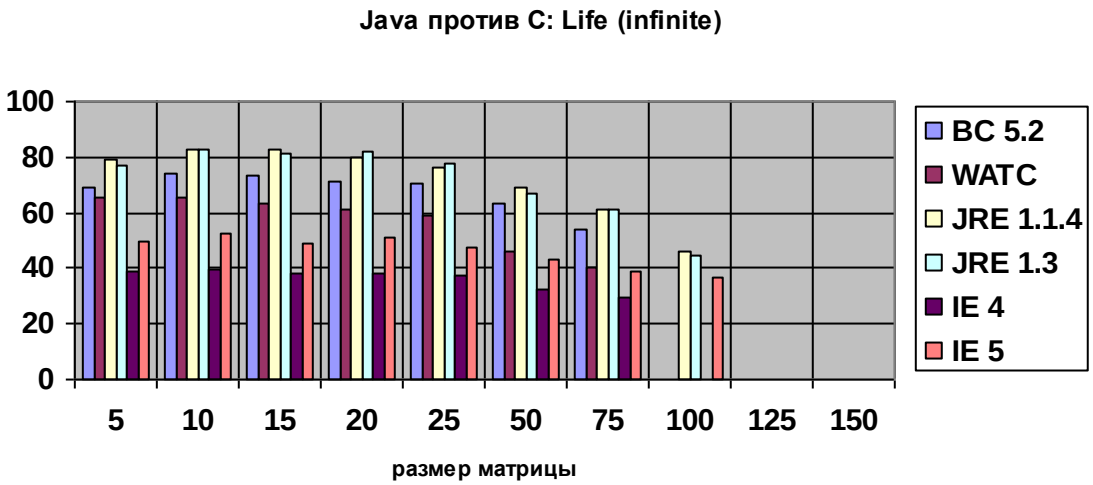
1. BC 5.2 – Borland C++ Compiler версии 5.2. Использовался единственный флаг оптимизации –4 для компиляции кода под 486 процессор.

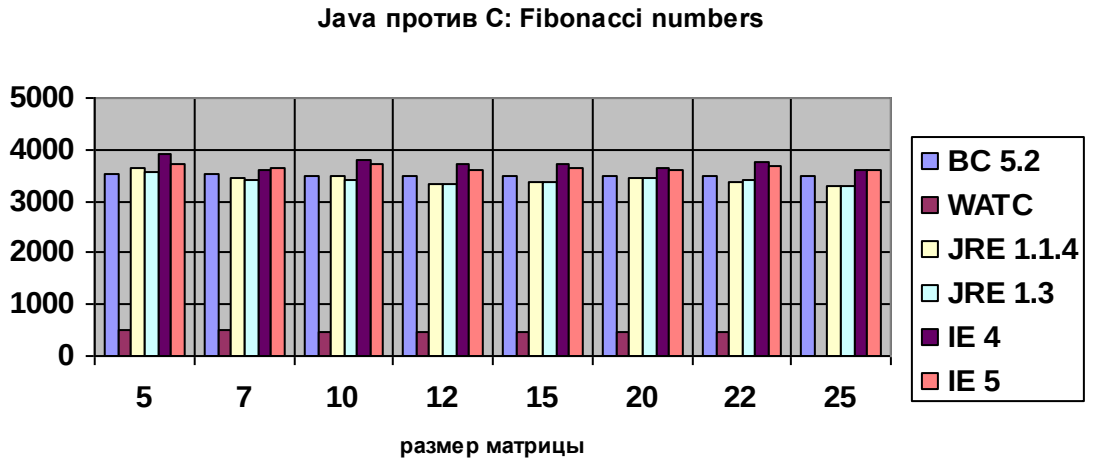
2. WATC – WATCOM C Compiler версии 10. Использовался -4r для компиляции кода под 486 процессор.
3. JRE 1.1.4 – Java Runtime Environment версии 1.1.4. Эта виртуальная машина используется в таких средах разработки, как Symantec Visual Cafe 2.1, а также в браузере Netscape Navigator 4.74.
4. JRE 1.3 – Java Runtime Environment версии 1.3. Виртуальная машина этой среды называется HotSpot и это самая последняя и наиболее продвинутая разработка от Sun.
5. IE 4 – Виртуальная машина Java, использующаяся в браузере Microsoft Internet Explorer 4.72.
6. IE 5 - Виртуальная машина Java, использующаяся в браузере Microsoft Internet Explorer 5.0.

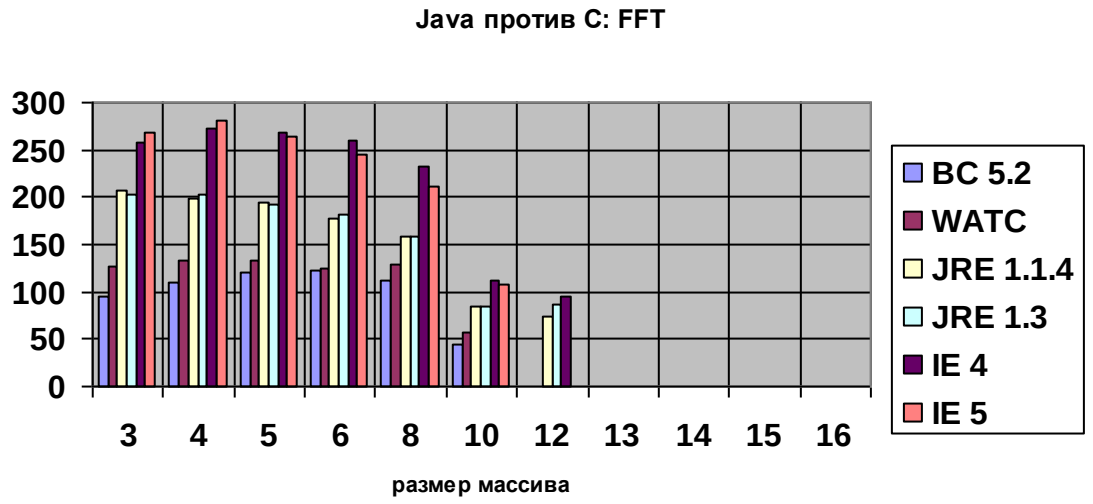
Результаты измерения производительности Java и C программ приведены на следующих диаграммах:






Java против C: Life (infinite)
100
80
60
40
20
0
5
10
15
20
25
50
75
100
125
150
размер матрицы
BC 5.2
WATC
JRE 1.1.4
JRE 1.3
IE 4
IE 5


Java против C: Fibonacci numbers
5000
4000
3000
2000
1000
0
5
7
10
12
15
20
22
25
размер матрицы
BC 5.2
WATC
JRE 1.1.4
JRE 1.3
IE 4
IE 5


Java против C: FFT
300
250
200
150
100
50
0
3
4
5
6
8
10
12
13
14
15
16
размер массива
BC 5.2
WATC
JRE 1.1.4
JRE 1.3
IE 4
IE 5

Выводы. По полученным результатам можно сделать вывод, что использование технологии Java значительно не сказывается на производительности системы. Более того, производительность Java-программ в некоторых случаях больше чем C-программ. Конечно не следует забывать, что проведенные тесты, по своей натуре, низкоуровневые и не могут служить предпосылками такой же производительности в приложениях реального мира. Сравнения больших Java и C/C++ программ с развитым графическим интерфейсом еще больше различаются, поскольку многое зависит от различных сред окружения.

3.6. Выводы по главе

1. На основе функциональной структуры (п. 2.3.5) программно реализован модуль минимизации ФАЛ.
2. На основе функциональной структуры (п. 2.4.3.) программно реализован модуль тестирования цифровых схем.
3. Разработан набор учебных цифровых схем различных комбинационных и последовательностных устройств (приложение 1).
4. Разработан набор контрольных тестовых заданий, используемых модулем тестирования, для поддержки изучения работы цифровых схем (приложение 2).
5. С помощью разработанного набора контрольных тестовых заданий проверена работоспособность модуля тестирования.
6. Программный код представлен в виде Java-классов и работоспособен на реализациях JVM начиная с версии 1.1. Это позволяет использовать его в интернет/интранет среде используя в качестве JVM: appletviwer из состава JDK версий 1.1 и выше; Internet Explorer версий 4.0 и выше; Netscape Navigator версий 4.5 и выше.
7. Использование Java-технологии для реализации программной части значительно не сказывается на производительности системы.

Заключение

В результате проделанной работы в направлении исследования и разработки средств поддержки изучения схемотехнического проектирования цифровых устройств были получены следующие результаты:

1. Сформулирован общий подход для разработки систем поддержки изучения цифровой схемотехники, функционирующих в телекоммуникационной среде и рассчитанных на поддержку процесса обучения.
2. Определены общие функциональные структуры средств поддержки изучения цифровой схемотехники в ТССМ.
3. Определены общие подходы и методы тестирования (контроля качества) моделей цифровых схем и устройств.
4. Определены общие программные структуры модулей минимизации ФАЛ и тестирования (контроля качества) цифровых схем, которые реализованы на языке Java.
5. Разработаны учебные модели различных цифровых устройств и тестовые задания для проверки их функционирования.
6. Полученные в результате эксперимента данные позволяют опровергнуть мнение о значительных потерях производительности для программ, выполняемых в среде JVM и заключить, что технология Java может активно использоваться при разработке интерактивных систем, в том числе ориентированных на использование в учебном процессе.

Использование изложенных идей в практической программной реализации и тестирование модулей поддержки изучения цифровой схемотехники в ТССМ показали, что данный подход является вполне адекватным и достаточно общим, хотя может и не универсальным. Реализованная система в целом отвечает поставленным перед ней требованиям. Очевидно, что для частичного или полного устранения недостатков и улучшения функциональности системы требуются дальнейшая работа в данном направлении.

Литература

1. Вишняков Ю.М., Родзин С.И. Интегрированная интеллектуальная система дистанционного образования. // Труды ДГТУ. – Донецк: ДГТУ. – 1999. Вып. (6). – С. 260-265.
2. Вишняков Ю.М., Родзин С.И. Информатизация наукоемких технологий обучения. // Интеллектуальные САПР. – Таганрог: ТРТУ. – 1998. Вып. 2(8). – С. 226-230.
3. Вишняков Ю.М., Родзин С.И. Виртуальная среда обучения: СВР + образовательные технологии. // Труды международной НПК «Компьютерные технологии: геометрическое моделирование и виртуальная реальность». – Таганрог: ТРТУ. – 1999. – С. 49-53.
4. Вишняков Ю.М., Родзин С.И. Интеллектуальный сервер международной лаборатории ELDIC. // Труды 9 международной НМК «Наукоемкие технологии образования». – Таганрог: ТРТУ. – 1999. – С. 68-69.
5. Головина Е.Ю., Чибизова Н.В. О построении интеллектуальной обучающей системы. // Теория и системы управления. – 1996. – N5. – С. 85-92.
6. Гориев В.Ф., Тарасов В.Б. Виртуальная кафедра – базовая единица вуза 21 века. // Сборник трудов «Проблемы регионального управления и инновационных процессов в образовании». – Таганрог: ТИУЭ. – 1999. – С. 29-30.
7. Родзин С.И. Отказоустойчивый протокол передачи и маршрутизации сообщений в многопроцессорных сетях с кольцевой архитектурой. // Труды международной НТК «Интеллектуальные многопроцессорные системы». – Таганрог: ТРТУ. – 1999. – С. 62-63.
8. Стефанюк В.Л. Теоретические аспекты разработки компьютерных систем обучения. – Саратов: СГУ. – 1995.
9. Тарасов В.Б. Системно-организационный подход в искусственном интеллекте. // Программные продукты и системы. – 1997. – N3. – С. 6-13.
10. Тихомиров В.Н., Солдаткин В.И., Лобачев С.Л. Виртуальная образовательная среда: предпосылки, принципы, организация. – М.: МЭСИ. – 1999.
11. Object Management Group: The Common Request Broker Architecture & Specification. Revision 2.0. – http://www.omg.org.
12. Яргер Р., Риз Дж., Кинг Т. MySQL и mSQL. Базы данных для небольших предприятий и Интернета – СПб: Символ-Плюс, 2000.
13. Кашицин В.П. Дистанционное обучение в высшей школе: модели и технологии. Москва. Центр информатизации Минобразования России. http://www.bspu.secna.ru/Journal/pi/pi_cash.html.
14. Баженова И.Ю. Язык программирования Java. - М.: ДИАЛОГ-МИФИ, 1997.
15. Орлик С. Обзор спецификации JDBC. // СУБД N3, 1997.

16. Вебер Д. Технология Java™ в подлиннике: пер. с англ. – СПб.: BHV – Санкт-Петербург, 1997.
17. Томас М. И др. Секреты программирования для Internet на Java. Перев. с англ. – СПб: Питер, 1997.
18. Майкл Морган. Java™ 2. Руководство разработчика.: пер. с англ. : Уч. пос. – М.: Издательский дом «Вильямс», 2000.
19. Томас Мейс, Нил Дж. Рубенкинг. ActiveX изнутри. PC Magazine, November 5, 1996, p. 267.
20. Ричард Хоффман, Роберт Патт-Корнер. «Капризы» управляющих элементов ActiveX и Java-апплетов. Сети и системы связи, N11, 1999.
21. Дон Маквитти. Java – достойный соперник или претендент в чемпионы. Сети и системы связи, N13, 1999.
22. Э. И. Гитис, Б. Л. Собкин, С. А. Илюшин и др. Автоматизация проектирования аналогово-цифровых устройств. // Под ред. Э. И. Гитиса. М.: Энергоатомиздат, 1987.
23. Барни К. Системы автоматизированного проектирования, ускоряющие моделирование благодаря применению реальных ИС. // Электроника. 1984. N 5. С. 3-4.
24. Негода В.Н. О контроле деятельности обучаемого в учебной САПР микропроцессорных систем В кн. «Вестник УлГТУ. Серия «Информационные технологии»». Ульяновск, УлГТУ, 2000.
25. Негода В.Н. О построении учебно-исследовательской системы функционально-логического моделирования микропроцессорных систем. - В кн. «Вестник УлГТУ. Серия «Информационные технологии»». Ульяновск, УлГТУ, 1998.
26. Соловов А.В. Проектирование компьютерных систем учебного назначения: Учебное пособие. Самара: СГАУ, 1995.
27. Афанасьев А.Н., Шишкин В.В. Методы и средства моделирования вычислительных устройств и систем: Учебное пособие. – Ульяновск: УГТУ, 1996.
28. Волков С.В., Михайлов В.Н. Телекоммуникационная система схемотехнического моделирования. - Материалы научно-методической конференции «Компьютерные технологии в высшем образовании», Ульяновск, УлГТУ, 1999. http://www.ulstu.ru/conf/NMK99/nmk99.html.
29. Колесников А.О., Негода В.Н. Встраивание процедур верификации знаний в учебную САПР МПС. – Труды международной конференции «Континуальные логико-алгебраические исчисления и нейроматематика в науке, технике и экономике», Том 2, Ульяновск, УлГТУ, 2001.
30. Советов Б.Я., Яковлев С.А. Моделирование систем: Учеб. Для вузов по спец. «Автоматизированные системы обработки информации и управления». - 2-е изд., перераб. и доп. - М.: Высш. шк., 1998.
31. Советов Б.Я. Информационная технология. - М.: Высшая школа, 1994.
32. Советов Б.Я., Яковлев С.А. Моделирование систем. - М.: Высшая школа, 1985.
33. Шеннон Р. Имитационное моделирование систем – Искусство и наука. - М.: Мир, 1978.

34. Ратанова О.В. Обучение контроллерам прерываний в телекоммуникационной среде. - Материалы научно-методической конференции «Компьютерные технологии в высшем образовании». Тезисы докладов научно-методической конференции, Ульяновск, УлГТУ, 1999, с 22-24. http://www.ulstu.ru/conf/NMK99/nmk99.html.
35. Т.Н.Гнитецкая. Интернет в учебном процессе. Дальневосточный государственный университет. http://ftp.nsu.ru/conf/nit/96/notasect/node93.html
36. Афонин С.Л. Обработка результатов логического моделирования цифровых устройств САПР P-CAD; Метод. указ. для студ. спец. 23.03, 23.01. - Ульяновск: УлГТУ, 1996.
37. Разевиг В.Д. Система P-CAD 8.5-8.7; Руководство пользователя . - М.: СОЛОН-Р, 1999.
38. Разевиг В. Д. Система схемотехнического моделирования MICRO-CAP V. - М.: СОЛОН, 1997.
39. Данилин Н.С., Нуров Ю.Л. Диагностика и контроль качества изделий цифровой микроэлектроники. - М.: Изд-во стандартов, 1991.
40. Кирьянов К.Г. Сигнатурный анализ: Методическое пособие. – Нижний Новгород: НГУ, 1999.
41. Пухальский Г.И., Новосельцева Т.Я. Цифровые устройства; Учеб. пособие для студ. техн. спец. вузов . - СПб.: Политехника, 1996.
42. Филиппов А.Г., Белкин О.С. Проектирование логических узлов ЭВМ. – М.: Сов. Радио, 1974.
43. Литвинов В.А. Система HLCCAD высокоуровневого проектирования цифровых устройств. Автоматизация проектирования дискретных систем: Материалы 3-й междунар. конф., 10-12 нояб. 1999, в 3 томах / Академия наук Беларуси, Институт технической кибернетики, Минск, 1999. Т.3, с. 154-159. http://nit.gsu.unibel.by/resources/articles/HLCCAD/HLCCAD_99_CADDD.doc.
44. Литвинов В.А. Система высокоуровневого проектирования цифровых устройств (HLCCAD - High Level Chip Computer-Aided Design). Труды международной конференции «Информационные технологии в бизнесе, образовании и науке», Минск, 1999, с. 179-182. http://nit.gsu.unibel.by/resources/articles/HLCCAD/HLCCAD_99_ITESB.doc
45. Долинский М.С., Литвинов В.А., Галатин А.В. Система высокоуровневого проектирования цифровых устройств HLCCAD. Минск, Электроника. No.10. 1998, с. 19-22. http://nit.gsu.unibel.by/resources/articles/HLCCAD/HLCCAD_98_Electronika.doc
46. Воробьев Н.В. Введение в булеву алгебру // Chip News. - 1997. http://www.chipnews.ru.
47. Воробьев Н.В. Формы представления и классификация функций алгебры логики // Chip News. - 1997. http://www.chipnews.ru.

48. Глушков В.М. Синтез цифровых автоматов. - М.: Физматгиз. 1962.
49. Лысиков Б.Г. Арифметические и логические основы цифровых автоматов: (Учебник для вузов по спец. «Электрон. вычисл. Машины»). - 2-изд., перераб. и доп. - Мн.: Высш. школа. 1980.
50. Ситников Ю.К., Таюрская Г.В. Система обучения диагностике с использованием компактных оценок. Тезисы докладов электронной конференции «Эффективность использования новых информационных технологий в учебном процессе». Ульяновск: УГУ, 2000.
51. Иванов М.А., Тышкевич В.Г., Зиборова М.Э., Кларин А.П. Техническое диагностирование цифровых устройств с использованием сигнатурного анализа. - М.: МИФИ, 1989.
52. Учебная система проектирования цифровых устройств на вентильных матрицах. Пензенский государственный университет. http://www.stup.ac.ru/WIN/SOFT/DESCRIPT/0001.ru.html.
53. Архангельский А.Я., PSpice и Design Center. В 2-х ч. Схемотехническое моделирование. Модели элементов. Макромоделирование. Учебное пособие. М.: МИФИ, 1996.
54. Советов Б.Я., Яковлев С.А. Моделирование систем: Учеб. Для вузов по спец. «Автоматизир. Системы обработки информ. И упр.». – 2-е изд., перераб. И доп. – М.: Высш. шк., 1998.
55. Мальцева Л.А. и др. Основы цифровой техники. – М.: Радио и связь, 1986.
56. Токхейм Р. Основы цифровой электроники: Пер. С англ. – М.: Мир, 1988.
57. Хакасский государственный университет. http://www.khsu.ru.
58. Пензенский государственный университет. http://www.stup.ac.ru.
59. Гомельский государственный университет. http://gsu.unibel.by.
60. Java Grande Benchmarking Project at Edinburgh Parallel Computing Centre. The University of Edinburgh. http://www.epcc.ed.ac.uk.
61. Проект JavaPower. Java против C. Версия Ace’s hardware. http://www.javapower.ru/java/vs/ace/ace1.htm.

Приложение 1. Набор учебных цифровых схем

Приводится исходный код 3-х цифровых схем из 15-ти разработанных.

```
//
// Цифровая схема 4-х разрядного вычитающего счетчика
//

//**********************************************************************
//   This scheme was created with Net schematics 1.00
//**********************************************************************
" Class : schematics.program.wire; Coordinates : 12,6,12,11; "
" Class : schematics.program.wire; Coordinates : 11,13,12,13; "
" Class : schematics.program.wire; Coordinates : 11,14,12,14; "
" Class : schematics.program.wire; Coordinates : 11,13,11,14; "
" Class : schematics.program.wire; Coordinates : 10,15,12,15; "
" Class : schematics.program.wire; Coordinates : 9,9,9,17; "
" Class : schematics.program.wire; Coordinates : 11,7,11,13; "
" Class : schematics.program.wire; Coordinates : 17,5,17,12; "
" Class : schematics.program.wire; Coordinates : 17,16,18,16; "
" Class : schematics.program.wire; Coordinates : 5,17,9,17; "
" Class : schematics.program.wire; Coordinates : 9,17,12,17; "
" Class : schematics.program.wire; Coordinates : 5,12,10,12; "
" Class : schematics.program.wire; Coordinates : 10,8,10,12; "
" Class : schematics.program.wire; Coordinates : 10,12,10,15; "
" Class : schematics.program.wire; Coordinates : 5,7,11,7; "
" Class : schematics.program.wire; Coordinates : 5,5,12,5; "
" Class : schematics.program.wire; Coordinates : 12,5,12,6; "
" Class : schematics.program.wire; Coordinates : 17,12,17,13; "
" Class : schematics.program.wire; Coordinates : 18,8,18,11; "
" Class : schematics.program.wire; Coordinates : 10,8,18,8; "
" Class : schematics.program.wire; Coordinates : 17,13,18,13; "
" Class : schematics.program.wire; Coordinates : 25,6,25,11; "
" Class : schematics.program.wire; Coordinates : 24,7,24,13; "
" Class : schematics.program.wire; Coordinates : 24,13,25,13; "
" Class : schematics.program.wire; Coordinates : 24,14,25,14; "
" Class : schematics.program.wire; Coordinates : 24,13,24,14; "
" Class : schematics.program.wire; Coordinates : 30,12,30,13; "
" Class : schematics.program.wire; Coordinates : 23,17,25,17; "
" Class : schematics.program.wire; Coordinates : 23,9,23,17; "
" Class : schematics.program.wire; Coordinates : 12,6,25,6; "
" Class : schematics.program.wire; Coordinates : 11,7,24,7; "
" Class : schematics.program.wire; Coordinates : 9,9,23,9; "
" Class : schematics.program.wire; Coordinates : 22,12,22,15; "
" Class : schematics.program.wire; Coordinates : 22,15,25,15; "
" Class : schematics.program.wire; Coordinates : 30,16,31,16; "
" Class : schematics.program.wire; Coordinates : 30,13,32,13; "
" Class : schematics.program.wire; Coordinates : 32,5,32,11; "
```

" Class : schematics.program.wire; Coordinates : 31,12,32,12; "
" Class : schematics.program.wire; Coordinates : 31,8,31,12; "
" Class : schematics.program.wire; Coordinates : 36,12,36,15; "
" Class : schematics.program.wire; Coordinates : 30,4,30,12; "
" Class : schematics.program.wire; Coordinates : 17,5,32,5; "
" Class : schematics.program.wire; Coordinates : 18,8,31,8; "
" Class : schematics.program.wire; Coordinates : 38,14,39,14; "
" Class : schematics.program.wire; Coordinates : 38,13,38,14; "
" Class : schematics.program.wire; Coordinates : 38,13,39,13; "
" Class : schematics.program.wire; Coordinates : 44,12,44,14; "
" Class : schematics.program.wire; Coordinates : 37,17,39,17; "
" Class : schematics.program.wire; Coordinates : 23,9,37,9; "
" Class : schematics.program.wire; Coordinates : 36,15,39,15; "
" Class : schematics.program.wire; Coordinates : 44,16,45,16; "
" Class : schematics.program.wire; Coordinates : 47,4,47,11; "
" Class : schematics.program.wire; Coordinates : 46,12,47,12; "
" Class : schematics.program.wire; Coordinates : 46,5,46,12; "
" Class : schematics.program.wire; Coordinates : 45,13,47,13; "
" Class : schematics.program.wire; Coordinates : 45,8,45,13; "
" Class : schematics.program.wire; Coordinates : 30,4,47,4; "
" Class : schematics.program.wire; Coordinates : 32,5,46,5; "
" Class : schematics.program.wire; Coordinates : 31,8,45,8; "
" Class : schematics.program.wire; Coordinates : 44,14,47,14; "
" Class : schematics.program.wire; Coordinates : 53,13,54,13; "
" Class : schematics.program.wire; Coordinates : 53,14,54,14; "
" Class : schematics.program.wire; Coordinates : 53,13,53,14; "
" Class : schematics.program.wire; Coordinates : 52,17,54,17; "
" Class : schematics.program.wire; Coordinates : 52,9,52,17; "
" Class : schematics.program.wire; Coordinates : 51,12,51,15; "
" Class : schematics.program.wire; Coordinates : 51,15,54,15; "
" Class : schematics.program.wire; Coordinates : 59,16,60,16; "
" Class : schematics.program.wire; Coordinates : 53,7,53,13; "
" Class : schematics.program.wire; Coordinates : 54,6,54,11; "
" Class : schematics.program.wire; Coordinates : 39,6,39,11; "
" Class : schematics.program.wire; Coordinates : 25,6,39,6; "
" Class : schematics.program.wire; Coordinates : 39,6,54,6; "
" Class : schematics.program.wire; Coordinates : 37,9,37,17; "
" Class : schematics.program.wire; Coordinates : 37,9,52,9; "
" Class : schematics.program.wire; Coordinates : 38,7,38,13; "
" Class : schematics.program.wire; Coordinates : 24,7,38,7; "
" Class : schematics.program.wire; Coordinates : 38,7,53,7; "
" Class : schematics.program.wire; Coordinates : 18,22,19,22; "
" Class : schematics.program.wire; Coordinates : 19,22,20,22; "
" Class : schematics.program.wire; Coordinates : 18,16,18,22; "
" Class : schematics.program.wire; Coordinates : 31,22,32,22; "
" Class : schematics.program.wire; Coordinates : 32,22,33,22; "
" Class : schematics.program.wire; Coordinates : 31,16,31,22; "
" Class : schematics.program.wire; Coordinates : 45,22,46,22; "
" Class : schematics.program.wire; Coordinates : 46,22,47,22; "
" Class : schematics.program.wire; Coordinates : 45,16,45,22; "

" Class : schematics.program.wire; Coordinates : 60,22,61,22; "
" Class : schematics.program.wire; Coordinates : 61,22,62,22; "
" Class : schematics.program.wire; Coordinates : 60,16,60,22; "
" Class : schematics.program.voltmeter; Coordinates : 5,2; Label : S; "
" Class : schematics.program.voltmeter; Coordinates : 5,14; Label : R; "
" Class : schematics.program.voltmeter; Coordinates : 5,9; Label : Synchro; "
" Class : schematics.program.voltmeter; Coordinates : 19,19; Label : Q0; "
" Class : schematics.program.voltmeter; Coordinates : 32,19; Label : Q1; "
" Class : schematics.program.voltmeter; Coordinates : 46,19; Label : Q2; "
" Class : schematics.program.voltmeter; Coordinates : 61,19; Label : Q3; "
" Class : schematics.elements.elements.digclock; Coordinates : 1,6; Parameters : Start value = 0.0, Start delay = 0.0, Ontime = 20.0, Offtime = 0.0; "
" Class : schematics.elements.elements.digclock; Coordinates : 1,4; Parameters : Start value = 1.0, Start delay = 0.0, Ontime = 20.0, Offtime = 0.0; Label : RES; "
" Class : schematics.elements.elements.digclock; Coordinates : 1,16; Parameters : Start value = 0.0, Start delay = 0.0, Ontime = 1000.0, Offtime = 20.0; Label : SET; "
" Class : schematics.elements.elements.digclock; Coordinates : 1,11; Parameters : Start value = 0.0, Start delay = 0.0, Ontime = 4.0, Offtime = 40.0; Label : TAKT; "
" Class : schematics.elements.k555.k555li2; Coordinates : 18,10; Parameters : delay = 12.0; "
" Class : schematics.elements.k555.k555li3; Coordinates : 32,10; Parameters : delay = 12.0; "
" Class : schematics.elements.k555.k555li6; Coordinates : 47,10; Parameters : delay = 12.0; "
" Class : schematics.elements.k555.k555tv9; Coordinates : 54,10; Parameters : delay = 30.0; "
" Class : schematics.elements.k555.k555tv9; Coordinates : 39,10; Parameters : delay = 30.0; "
" Class : schematics.elements.k555.k555tv9; Coordinates : 25,10; Parameters : delay = 30.0; "
" Class : schematics.elements.k555.k555tv9; Coordinates : 12,10; Parameters : delay = 30.0; "
//
// Цифровая схема 8-ми разрядного регистра с последовательной загрузкой со
// сдвигом влево
//

//***
// This scheme was created with Net schematics 1.00
//***
" Class : schematics.program.wire; Coordinates : 5,6,6,6; "
" Class : schematics.program.wire; Coordinates : 5,12,6,12; "
" Class : schematics.program.wire; Coordinates : 5,18,6,18; "
" Class : schematics.program.wire; Coordinates : 5,24,6,24; "
" Class : schematics.program.voltmeter; Coordinates : 6,3; Label : SET; "
" Class : schematics.program.voltmeter; Coordinates : 6,9; Label : DAT; "
" Class : schematics.program.voltmeter; Coordinates : 6,15; Label : TAKT; "
" Class : schematics.program.voltmeter; Coordinates : 6,21; Label : RES; "
" Class : schematics.program.wire; Coordinates : 6,18,10,18; "
" Class : schematics.program.wire; Coordinates : 10,13,10,18; "

" Class : schematics.program.wire; Coordinates : 10,13,11,13; "
" Class : schematics.program.wire; Coordinates : 6,12,11,12; "
" Class : schematics.program.wire; Coordinates : 11,15,11,24; "
" Class : schematics.program.wire; Coordinates : 6,24,11,24; "
" Class : schematics.program.wire; Coordinates : 10,18,17,18; "
" Class : schematics.program.wire; Coordinates : 11,24,18,24; "
" Class : schematics.program.wire; Coordinates : 16,5,16,11; "
" Class : schematics.program.wire; Coordinates : 16,11,16,12; "
" Class : schematics.program.wire; Coordinates : 17,13,18,13; "
" Class : schematics.program.wire; Coordinates : 17,13,17,18; "
" Class : schematics.program.wire; Coordinates : 18,15,18,24; "
" Class : schematics.program.wire; Coordinates : 18,6,18,10; "
" Class : schematics.program.wire; Coordinates : 16,12,18,12; "
" Class : schematics.program.wire; Coordinates : 23,5,23,11; "
" Class : schematics.program.wire; Coordinates : 23,11,23,12; "
" Class : schematics.program.wire; Coordinates : 24,13,25,13; "
" Class : schematics.program.wire; Coordinates : 24,13,24,18; "
" Class : schematics.program.wire; Coordinates : 25,15,25,24; "
" Class : schematics.program.wire; Coordinates : 25,6,25,10; "
" Class : schematics.program.wire; Coordinates : 23,12,25,12; "
" Class : schematics.program.wire; Coordinates : 30,5,30,11; "
" Class : schematics.program.wire; Coordinates : 30,11,30,12; "
" Class : schematics.program.wire; Coordinates : 31,13,32,13; "
" Class : schematics.program.wire; Coordinates : 31,13,31,18; "
" Class : schematics.program.wire; Coordinates : 32,15,32,24; "
" Class : schematics.program.wire; Coordinates : 32,6,32,10; "
" Class : schematics.program.wire; Coordinates : 30,12,32,12; "
" Class : schematics.program.wire; Coordinates : 17,18,24,18; "
" Class : schematics.program.wire; Coordinates : 18,24,25,24; "
" Class : schematics.program.wire; Coordinates : 18,6,25,6; "
" Class : schematics.program.wire; Coordinates : 37,5,37,11; "
" Class : schematics.program.wire; Coordinates : 37,11,37,12; "
" Class : schematics.program.wire; Coordinates : 38,13,39,13; "
" Class : schematics.program.wire; Coordinates : 38,13,38,18; "
" Class : schematics.program.wire; Coordinates : 39,15,39,24; "
" Class : schematics.program.wire; Coordinates : 39,6,39,10; "
" Class : schematics.program.wire; Coordinates : 37,12,39,12; "
" Class : schematics.program.wire; Coordinates : 24,18,31,18; "
" Class : schematics.program.wire; Coordinates : 25,24,32,24; "
" Class : schematics.program.wire; Coordinates : 25,6,32,6; "
" Class : schematics.program.wire; Coordinates : 44,5,44,11; "
" Class : schematics.program.wire; Coordinates : 44,11,44,12; "
" Class : schematics.program.wire; Coordinates : 45,13,46,13; "
" Class : schematics.program.wire; Coordinates : 45,13,45,18; "
" Class : schematics.program.wire; Coordinates : 46,15,46,24; "
" Class : schematics.program.wire; Coordinates : 46,6,46,10; "
" Class : schematics.program.wire; Coordinates : 44,12,46,12; "
" Class : schematics.program.wire; Coordinates : 31,18,38,18; "
" Class : schematics.program.wire; Coordinates : 32,24,39,24; "
" Class : schematics.program.wire; Coordinates : 32,6,39,6; "

" Class : schematics.program.wire; Coordinates : 51,5,51,11; "
" Class : schematics.program.wire; Coordinates : 51,11,51,12; "
" Class : schematics.program.wire; Coordinates : 52,13,53,13; "
" Class : schematics.program.wire; Coordinates : 52,13,52,18; "
" Class : schematics.program.wire; Coordinates : 53,15,53,24; "
" Class : schematics.program.wire; Coordinates : 53,6,53,10; "
" Class : schematics.program.wire; Coordinates : 51,12,53,12; "
" Class : schematics.program.wire; Coordinates : 38,18,45,18; "
" Class : schematics.program.wire; Coordinates : 39,24,46,24; "
" Class : schematics.program.wire; Coordinates : 39,6,46,6; "
" Class : schematics.program.wire; Coordinates : 58,5,58,11; "
" Class : schematics.program.wire; Coordinates : 58,11,58,12; "
" Class : schematics.program.wire; Coordinates : 59,13,60,13; "
" Class : schematics.program.wire; Coordinates : 59,13,59,18; "
" Class : schematics.program.wire; Coordinates : 60,6,60,10; "
" Class : schematics.program.wire; Coordinates : 58,12,60,12; "
" Class : schematics.program.wire; Coordinates : 45,18,52,18; "
" Class : schematics.program.wire; Coordinates : 46,24,53,24; "
" Class : schematics.program.wire; Coordinates : 46,6,53,6; "
" Class : schematics.program.wire; Coordinates : 52,18,59,18; "
" Class : schematics.program.wire; Coordinates : 53,24,60,24; "
" Class : schematics.program.wire; Coordinates : 53,6,60,6; "
" Class : schematics.program.wire; Coordinates : 60,15,60,24; "
" Class : schematics.program.wire; Coordinates : 65,5,65,11; "
" Class : schematics.program.wire; Coordinates : 11,6,11,10; "
" Class : schematics.program.wire; Coordinates : 6,6,11,6; "
" Class : schematics.program.wire; Coordinates : 11,6,18,6; "
" Class : schematics.program.voltmeter; Coordinates : 17,2; Label : Q0; "
" Class : schematics.program.voltmeter; Coordinates : 24,2; Label : Q1; "
" Class : schematics.program.wire; Coordinates : 16,5,17,5; "
" Class : schematics.program.wire; Coordinates : 17,5,18,5; "
" Class : schematics.program.voltmeter; Coordinates : 31,2; Label : Q2; "
" Class : schematics.program.wire; Coordinates : 23,5,24,5; "
" Class : schematics.program.wire; Coordinates : 24,5,25,5; "
" Class : schematics.program.voltmeter; Coordinates : 38,2; Label : Q3; "
" Class : schematics.program.wire; Coordinates : 30,5,31,5; "
" Class : schematics.program.wire; Coordinates : 31,5,32,5; "
" Class : schematics.program.voltmeter; Coordinates : 45,2; Label : Q4; "
" Class : schematics.program.wire; Coordinates : 37,5,38,5; "
" Class : schematics.program.wire; Coordinates : 38,5,39,5; "
" Class : schematics.program.voltmeter; Coordinates : 52,2; Label : Q5; "
" Class : schematics.program.wire; Coordinates : 44,5,45,5; "
" Class : schematics.program.wire; Coordinates : 45,5,46,5; "
" Class : schematics.program.wire; Coordinates : 51,5,52,5; "
" Class : schematics.program.wire; Coordinates : 52,5,53,5; "
" Class : schematics.program.voltmeter; Coordinates : 59,2; Label : Q6; "
" Class : schematics.program.voltmeter; Coordinates : 66,2; Label : Q7; "
" Class : schematics.program.wire; Coordinates : 58,5,59,5; "
" Class : schematics.program.wire; Coordinates : 59,5,60,5; "
" Class : schematics.program.wire; Coordinates : 65,5,66,5; "

" Class : schematics.program.wire; Coordinates : 66,5,67,5; "
" Class : schematics.elements.k155.k155tm2; Coordinates : 11,9; Parameters : delay = 4.0; "
" Class : schematics.elements.k155.k155tm2; Coordinates : 18,9; Parameters : delay = 4.0; "
" Class : schematics.elements.k155.k155tm2; Coordinates : 25,9; Parameters : delay = 4.0; "
" Class : schematics.elements.k155.k155tm2; Coordinates : 32,9; Parameters : delay = 4.0; "
" Class : schematics.elements.k155.k155tm2; Coordinates : 39,9; Parameters : delay = 4.0; "
" Class : schematics.elements.k155.k155tm2; Coordinates : 46,9; Parameters : delay = 4.0; "
" Class : schematics.elements.k155.k155tm2; Coordinates : 53,9; Parameters : delay = 4.0; "
" Class : schematics.elements.k155.k155tm2; Coordinates : 60,9; Parameters : delay = 4.0; "
" Class : schematics.elements.elements.digclock; Coordinates : 1,5; Parameters : Start value = 1.0, Start delay = 0.0, Ontime = 135.0, Offtime = 5.0; Label : SET; "
" Class : schematics.elements.elements.digclock; Coordinates : 1,11; Parameters : Start value = 1.0, Start delay = 0.0, Ontime = 15.0, Offtime = 50.0; Label : DAT; "
" Class : schematics.elements.elements.digclock; Coordinates : 1,17; Parameters : Start value = 1.0, Start delay = 0.0, Ontime = 3.0, Offtime = 4.0; Label : TAKT; "
" Class : schematics.elements.elements.digclock; Coordinates : 1,23; Parameters : Start value = 0.0, Start delay = 0.0, Ontime = 115.0, Offtime = 5.0; Label : RES; "

//
// Цифровая схема Т-триггера
//

//**
// This scheme was created with Net schematics 1.00
//**
" Class : schematics.program.wire; Coordinates : 15,18,16,18; "
" Class : schematics.program.wire; Coordinates : 15,14,15,18; "
" Class : schematics.program.wire; Coordinates : 15,19,15,22; "
" Class : schematics.program.wire; Coordinates : 20,18,20,21; "
" Class : schematics.program.wire; Coordinates : 16,21,20,21; "
" Class : schematics.program.wire; Coordinates : 15,22,20,22; "
" Class : schematics.program.wire; Coordinates : 16,21,16,24; "
" Class : schematics.program.wire; Coordinates : 16,26,16,29; "
" Class : schematics.program.wire; Coordinates : 14,28,20,28; "
" Class : schematics.program.wire; Coordinates : 20,25,20,28; "
" Class : schematics.program.wire; Coordinates : 15,19,16,19; "
" Class : schematics.program.wire; Coordinates : 16,14,21,14; "
" Class : schematics.program.wire; Coordinates : 14,3,16,3; "
" Class : schematics.program.wire; Coordinates : 16,5,16,7; "
" Class : schematics.program.wire; Coordinates : 16,7,21,7; "
" Class : schematics.program.wire; Coordinates : 20,4,20,8; "
" Class : schematics.program.wire; Coordinates : 16,8,20,8; "
" Class : schematics.program.wire; Coordinates : 16,8,16,10; "
" Class : schematics.program.wire; Coordinates : 20,11,21,11; "
" Class : schematics.program.wire; Coordinates : 21,11,21,14; "
" Class : schematics.program.wire; Coordinates : 15,12,16,12; "
" Class : schematics.program.wire; Coordinates : 14,3,14,28; "
" Class : schematics.program.wire; Coordinates : 16,14,16,17; "
" Class : schematics.program.wire; Coordinates : 15,12,15,14; "
" Class : schematics.program.wire; Coordinates : 20,22,20,25; "

" Class : schematics.program.wire; Coordinates : 13,14,15,14; "
" Class : schematics.elements.k155.k155la3; Coordinates : 16,2; Parameters : delay = 10.0; "
" Class : schematics.elements.k155.k155la3; Coordinates : 16,9; Parameters : delay = 10.0; "
" Class : schematics.elements.k155.k155la4; Coordinates : 16,16; Parameters : delay = 10.0;
"
" Class : schematics.elements.k155.k155la3; Coordinates : 16,23; Parameters : delay = 10.0;
"
" Class : schematics.program.wire; Coordinates : 23,1,23,16; "
" Class : schematics.program.wire; Coordinates : 9,1,23,1; "
" Class : schematics.program.wire; Coordinates : 33,16,33,29; "
" Class : schematics.program.wire; Coordinates : 25,13,33,13; "
" Class : schematics.program.wire; Coordinates : 25,11,25,13; "
" Class : schematics.program.wire; Coordinates : 33,13,33,16; "
" Class : schematics.program.wire; Coordinates : 29,10,29,15; "
" Class : schematics.program.wire; Coordinates : 29,10,34,10; "
" Class : schematics.program.wire; Coordinates : 34,10,35,10; "
" Class : schematics.program.wire; Coordinates : 33,16,34,16; "
" Class : schematics.program.wire; Coordinates : 34,16,35,16; "
" Class : schematics.elements.k155.k155la3; Coordinates : 25,8; Parameters : delay = 10.0; "
" Class : schematics.elements.k155.k155la3; Coordinates : 29,14; Parameters : delay = 10.0;
"
" Class : schematics.elements.k155.k155li1; Coordinates : 25,15; Parameters : delay = 15.0;
"
" Class : schematics.program.wire; Coordinates : 16,29,33,29; "
" Class : schematics.program.wire; Coordinates : 21,9,25,9; "
" Class : schematics.program.wire; Coordinates : 21,7,21,9; "
" Class : schematics.program.wire; Coordinates : 21,9,21,11; "
" Class : schematics.program.wire; Coordinates : 23,16,25,16; "
" Class : schematics.program.wire; Coordinates : 20,18,25,18; "
" Class : schematics.elements.elements.digclock; Coordinates : 1,9; Parameters : Start value = 0.0, Start delay = 0.0, Ontime = 1000.0, Offtime = 40.0; Label : RES; "
" Class : schematics.elements.k155.k155li1; Coordinates : 9,12; Parameters : delay = 15.0; "
" Class : schematics.elements.elements.digclock; Coordinates : 1,14; Parameters : Start value = 0.0, Start delay = 0.0, Ontime = 50.0, Offtime = 70.0; Label : TAKT; "
" Class : schematics.program.wire; Coordinates : 5,15,9,15; "
" Class : schematics.program.wire; Coordinates : 5,10,9,10; "
" Class : schematics.program.wire; Coordinates : 9,1,9,10; "
" Class : schematics.program.wire; Coordinates : 9,10,9,13; "
" Class : schematics.elements.elements.textinfo; Coordinates : 5,30; Label : This is T-trigger example. This scheme was created by SerVo.; "
" Class : schematics.program.voltmeter; Coordinates : 5,7; Label : RES; "
" Class : schematics.program.voltmeter; Coordinates : 5,12; Label : TAKT; "
" Class : schematics.program.voltmeter; Coordinates : 34,7; Label : Q; "
" Class : schematics.program.voltmeter; Coordinates : 34,13; Label : NotQ; "

Приложение 2. Набор контрольных тестовых заданий для проверки работоспособности учебных цифровых схем

Приводится исходный код 3-х контрольных тестовых заданий из 15-ти разработанных.

```
#
#   Название,   класс   и
формулировка задания
#

title=4-х          разрядный
вычитающий счетчик
class=Счетчики
description=Спроектировать
4-х разрядный вычитающий
счетчик.      В      схеме
необходимо использовать не
менее 3-х генераторов и 4-
х маркеров. Генераторы:
RES - по 0 фронту счетчик
должен обнуляться; SET -
по 0 фронту счетчик должен
заполняться      единицами;
ТАКТ     -     осуществляет
тактирование. Маркеры: Q0,
Q1,   Q2,   Q3   -   должны
отражать          состояние
счетчика.

#
#       Генераторы       и
передаваемые           ими
управляющие воздействия
#

digclock.1.label=RES
digclock.1.time.1=1700
digclock.1.state.1=1
digclock.1.time.2=100
digclock.1.state.2=0

digclock.2.label=SET
digclock.2.time.1=100
digclock.2.state.1=0
digclock.2.time.2=1700
digclock.2.state.2=1

digclock.3.label=ТАКТ
digclock.3.time.1=100
digclock.3.state.1=0
digclock.3.time.2=1
digclock.3.state.2=1
digclock.3.time.3=99
digclock.3.state.3=0
digclock.3.time.4=1
digclock.3.state.4=1
digclock.3.time.5=99
digclock.3.state.5=0
digclock.3.time.6=1
digclock.3.state.6=1
digclock.3.time.7=99
digclock.3.state.7=0
digclock.3.time.8=1
digclock.3.state.8=1
digclock.3.time.9=99
digclock.3.state.9=0
digclock.3.time.10=1
digclock.3.state.10=1
digclock.3.time.11=99
digclock.3.state.11=0
digclock.3.time.12=1
digclock.3.state.12=1
digclock.3.time.13=99
digclock.3.state.13=0
digclock.3.time.14=1
digclock.3.state.14=1
digclock.3.time.15=99
digclock.3.state.15=0
digclock.3.time.16=1
digclock.3.state.16=1
digclock.3.time.17=99
digclock.3.state.17=0
digclock.3.time.18=1
digclock.3.state.18=1
digclock.3.time.19=99
digclock.3.state.19=0
digclock.3.time.20=1
digclock.3.state.20=1
digclock.3.time.21=99
```

```
digclock.3.state.21=0
digclock.3.time.22=1
digclock.3.state.22=1
digclock.3.time.23=99
digclock.3.state.23=0
digclock.3.time.24=1
digclock.3.state.24=1
digclock.3.time.25=99
digclock.3.state.25=0
digclock.3.time.26=1
digclock.3.state.26=1
digclock.3.time.27=99
digclock.3.state.27=0
digclock.3.time.28=1
digclock.3.state.28=1
digclock.3.time.29=99
digclock.3.state.29=0
digclock.3.time.30=1
digclock.3.state.30=1
digclock.3.time.31=99
digclock.3.state.31=0
digclock.3.time.32=1
digclock.3.state.32=1
digclock.3.time.33=199
digclock.3.state.33=0

#
#   Маркеры   и   ожидаемые
значения    на    тестовых
интервалах
#

marker.1.label=Q0
marker.1.time.1=100
marker.1.state.1=1
marker.1.time.2=100
marker.1.state.2=0
marker.1.time.3=100
marker.1.state.3=1
marker.1.time.4=100
marker.1.state.4=0
marker.1.time.5=100
marker.1.state.5=1
marker.1.time.6=100
marker.1.state.6=0
marker.1.time.7=100
marker.1.state.7=1
marker.1.time.8=100
marker.1.state.8=0
marker.1.time.9=100
marker.1.state.9=1
marker.1.time.10=100
marker.1.state.10=0
marker.1.time.11=100
marker.1.state.11=1
marker.1.time.12=100
marker.1.state.12=0
marker.1.time.13=100
marker.1.state.13=1
marker.1.time.14=100
marker.1.state.14=0
marker.1.time.15=100
marker.1.state.15=1
marker.1.time.16=100
marker.1.state.16=0
marker.1.time.17=100
marker.1.state.17=1
marker.1.time.18=100
marker.1.state.18=0

marker.2.label=Q1
marker.2.time.1=100
marker.2.state.1=1
marker.2.time.2=100
marker.2.state.2=1
marker.2.time.3=100
marker.2.state.3=0
marker.2.time.4=100
marker.2.state.4=0
marker.2.time.5=100
marker.2.state.5=1
marker.2.time.6=100
marker.2.state.6=1
marker.2.time.7=100
marker.2.state.7=0
marker.2.time.8=100
marker.2.state.8=0
marker.2.time.9=100
marker.2.state.9=1
marker.2.time.10=100
marker.2.state.10=1
marker.2.time.11=100
marker.2.state.11=0
marker.2.time.12=100
marker.2.state.12=0
marker.2.time.13=100
marker.2.state.13=1
marker.2.time.14=100
marker.2.state.14=1
marker.2.time.15=100
marker.2.state.15=0
marker.2.time.16=100
```

```
marker.2.state.16=0
marker.2.time.17=100
marker.2.state.17=1
marker.2.time.18=100
marker.2.state.18=0

marker.3.label=Q2
marker.3.time.1=100
marker.3.state.1=1
marker.3.time.2=100
marker.3.state.2=1
marker.3.time.3=100
marker.3.state.3=1
marker.3.time.4=100
marker.3.state.4=1
marker.3.time.5=100
marker.3.state.5=0
marker.3.time.6=100
marker.3.state.6=0
marker.3.time.7=100
marker.3.state.7=0
marker.3.time.8=100
marker.3.state.8=0
marker.3.time.9=100
marker.3.state.9=1
marker.3.time.10=100
marker.3.state.10=1
marker.3.time.11=100
marker.3.state.11=1
marker.3.time.12=100
marker.3.state.12=1
marker.3.time.13=100
marker.3.state.13=0
marker.3.time.14=100
marker.3.state.14=0
marker.3.time.15=100
marker.3.state.15=0
marker.3.time.16=100
marker.3.state.16=0
marker.3.time.17=100
marker.3.state.17=1
marker.3.time.18=100
marker.3.state.18=0

marker.4.label=Q3
marker.4.time.1=100
marker.4.state.1=1
marker.4.time.2=100
marker.4.state.2=1
marker.4.time.3=100
marker.4.state.3=1
marker.4.time.4=100
marker.4.state.4=1
marker.4.time.5=100
marker.4.state.5=1
marker.4.time.6=100
marker.4.state.6=1
marker.4.time.7=100
marker.4.state.7=1
marker.4.time.8=100
marker.4.state.8=1
marker.4.time.9=100
marker.4.state.9=0
marker.4.time.10=100
marker.4.state.10=0
marker.4.time.11=100
marker.4.state.11=0
marker.4.time.12=100
marker.4.state.12=0
marker.4.time.13=100
marker.4.state.13=0
marker.4.time.14=100
marker.4.state.14=0
marker.4.time.15=100
marker.4.state.15=0
marker.4.time.16=100
marker.4.state.16=0
marker.4.time.17=100
marker.4.state.17=1
marker.4.time.18=100
marker.4.state.18=0
#
#   Название, класс и
формулировка задания
#

title=8-ми разрядный
регистр с последовательной
загрузкой со сдвигом влево
class=Регистры
description=Спроектировать
8-ми разрядный регистр с
последовательной загрузкой
со сдвигом влево. В схеме
необходимо использовать не
менее 4 генераторов и 8
маркеров. Генераторы: RES
- по 0 фронту регистр
должен обнуляться; SET -
по 0 фронту регистр должен
заполняться единицами; DAT
- бит вдвигаемый при
```

```
очередном такте; ТАКТ -
осуществляет тактирование.
Маркеры: Q0, Q1, Q2; Q3;
Q4; Q5; Q6; Q7 - должны
отражать состояние
регистра.

#
#       Генераторы       и
передаваемые           ими
управляющие воздействия
#

digclock.1.label=RES
digclock.1.time.1=100
digclock.1.state.1=0
digclock.1.time.2=1000
digclock.1.state.2=1
digclock.1.time.3=100
digclock.1.state.3=0

digclock.2.label=SET
digclock.2.time.1=1000
digclock.2.state.1=1
digclock.2.time.2=100
digclock.2.state.2=0
digclock.2.time.3=100
digclock.2.state.3=1

digclock.3.label=DAT
digclock.3.time.1=300
digclock.3.state.1=1
digclock.3.time.2=100
digclock.3.state.2=0
digclock.3.time.3=100
digclock.3.state.3=1
digclock.3.time.4=700
digclock.3.state.4=0

digclock.4.label=TAKT
digclock.4.time.1=100
digclock.4.state.1=0
digclock.4.time.2=1
digclock.4.state.2=1
digclock.4.time.3=99
digclock.4.state.3=0
digclock.4.time.4=1
digclock.4.state.4=1
digclock.4.time.5=99
digclock.4.state.5=0
digclock.4.time.6=1
digclock.4.state.6=1
digclock.4.time.7=99
digclock.4.state.7=0
digclock.4.time.8=1
digclock.4.state.8=1
digclock.4.time.9=99
digclock.4.state.9=0
digclock.4.time.10=1
digclock.4.state.10=1
digclock.4.time.11=99
digclock.4.state.11=0
digclock.4.time.12=1
digclock.4.state.12=1
digclock.4.time.13=99
digclock.4.state.13=0
digclock.4.time.14=1
digclock.4.state.14=1
digclock.4.time.15=99
digclock.4.state.15=0
digclock.4.time.16=1
digclock.4.state.16=1
digclock.4.time.17=99
digclock.4.state.17=0
digclock.4.time.18=1
digclock.4.state.18=1
digclock.4.time.19=99
digclock.4.state.19=0
digclock.4.time.20=100
digclock.4.state.20=0
digclock.4.time.21=100
digclock.4.state.21=0

#
#  Маркеры  и  ожидаемые
значения    на    тестовых
интервалах
#

marker.1.label=Q0
marker.1.time.1=100
marker.1.state.1=0
marker.1.time.2=100
marker.1.state.2=1
marker.1.time.3=100
marker.1.state.3=1
marker.1.time.4=100
marker.1.state.4=0
marker.1.time.5=100
marker.1.state.5=1
marker.1.time.6=100
marker.1.state.6=0
```

```
marker.1.time.7=100
marker.1.state.7=0
marker.1.time.8=100
marker.1.state.8=0
marker.1.time.9=100
marker.1.state.9=0
marker.1.time.10=100
marker.1.state.10=0
marker.1.time.11=100
marker.1.state.11=1
marker.1.time.12=100
marker.1.state.12=0

marker.2.label=Q1
marker.2.time.1=100
marker.2.state.1=0
marker.2.time.2=100
marker.2.state.2=0
marker.2.time.3=100
marker.2.state.3=1
marker.2.time.4=100
marker.2.state.4=1
marker.2.time.5=100
marker.2.state.5=0
marker.2.time.6=100
marker.2.state.6=1
marker.2.time.7=100
marker.2.state.7=0
marker.2.time.8=100
marker.2.state.8=0
marker.2.time.9=100
marker.2.state.9=0
marker.2.time.10=100
marker.2.state.10=0
marker.2.time.11=100
marker.2.state.11=1
marker.2.time.12=100
marker.2.state.12=0

marker.3.label=Q2
marker.3.time.1=100
marker.3.state.1=0
marker.3.time.2=100
marker.3.state.2=0
marker.3.time.3=100
marker.3.state.3=0
marker.3.time.4=100
marker.3.state.4=1
marker.3.time.5=100
marker.3.state.5=1
marker.3.time.6=100
marker.3.state.6=0
marker.3.time.7=100
marker.3.state.7=1
marker.3.time.8=100
marker.3.state.8=0
marker.3.time.9=100
marker.3.state.9=0
marker.3.time.10=100
marker.3.state.10=0
marker.3.time.11=100
marker.3.state.11=1
marker.3.time.12=100
marker.3.state.12=0

marker.4.label=Q3
marker.4.time.1=100
marker.4.state.1=0
marker.4.time.2=100
marker.4.state.2=0
marker.4.time.3=100
marker.4.state.3=0
marker.4.time.4=100
marker.4.state.4=0
marker.4.time.5=100
marker.4.state.5=1
marker.4.time.6=100
marker.4.state.6=1
marker.4.time.7=100
marker.4.state.7=0
marker.4.time.8=100
marker.4.state.8=1
marker.4.time.9=100
marker.4.state.9=0
marker.4.time.10=100
marker.4.state.10=0
marker.4.time.11=100
marker.4.state.11=1
marker.4.time.12=100
marker.4.state.12=0

marker.5.label=Q4
marker.5.time.1=100
marker.5.state.1=0
marker.5.time.2=100
marker.5.state.2=0
marker.5.time.3=100
marker.5.state.3=0
marker.5.time.4=100
marker.5.state.4=0
marker.5.time.5=100
marker.5.state.5=0
```

```
marker.5.time.6=100
marker.5.state.6=1
marker.5.time.7=100
marker.5.state.7=1
marker.5.time.8=100
marker.5.state.8=0
marker.5.time.9=100
marker.5.state.9=1
marker.5.time.10=100
marker.5.state.10=0
marker.5.time.11=100
marker.5.state.11=1
marker.5.time.12=100
marker.5.state.12=0

marker.6.label=Q5
marker.6.time.1=100
marker.6.state.1=0
marker.6.time.2=100
marker.6.state.2=0
marker.6.time.3=100
marker.6.state.3=0
marker.6.time.4=100
marker.6.state.4=0
marker.6.time.5=100
marker.6.state.5=0
marker.6.time.6=100
marker.6.state.6=0
marker.6.time.7=100
marker.6.state.7=1
marker.6.time.8=100
marker.6.state.8=1
marker.6.time.9=100
marker.6.state.9=0
marker.6.time.10=100
marker.6.state.10=1
marker.6.time.11=100
marker.6.state.11=1
marker.6.time.12=100
marker.6.state.12=0

marker.7.label=Q6
marker.7.time.1=100
marker.7.state.1=0
marker.7.time.2=100
marker.7.state.2=0
marker.7.time.3=100
marker.7.state.3=0
marker.7.time.4=100
marker.7.state.4=0
marker.7.time.5=100
marker.7.state.5=0
marker.7.time.6=100
marker.7.state.6=0
marker.7.time.7=100
marker.7.state.7=0
marker.7.time.8=100
marker.7.state.8=1
marker.7.time.9=100
marker.7.state.9=1
marker.7.time.10=100
marker.7.state.10=0
marker.7.time.11=100
marker.7.state.11=1
marker.7.time.12=100
marker.7.state.12=0

marker.8.label=Q7
marker.8.time.1=100
marker.8.state.1=0
marker.8.time.2=100
marker.8.state.2=0
marker.8.time.3=100
marker.8.state.3=0
marker.8.time.4=100
marker.8.state.4=0
marker.8.time.5=100
marker.8.state.5=0
marker.8.time.6=100
marker.8.state.6=0
marker.8.time.7=100
marker.8.state.7=0
marker.8.time.8=100
marker.8.state.8=0
marker.8.time.9=100
marker.8.state.9=1
marker.8.time.10=100
marker.8.state.10=1
marker.8.time.11=100
marker.8.state.11=1
marker.8.time.12=100
marker.8.state.12=0
#
#   Название,   класс   и
формулировка задания
#

title=T-триггер
class=Триггеры
description=Спроектировать
T-триггер, переключающийся
по   1-фронту.   В   схеме
```

```
необходимо использовать не менее 2-х генераторов и 2 маркеров. Генераторы: RES - по 0 уровню триггер должен находиться в сброшенном состоянии; ТАКТ - осуществляет тактирование. Маркеры: Q, NOTQ - должны отражать состояние триггера.

#
#       Генераторы и передаваемые ими управляющие воздействия
#

digclock.1.label=RES
digclock.1.time.1=200
digclock.1.state.1=0
digclock.1.time.2=400
digclock.1.state.2=1
digclock.1.time.3=200
digclock.1.state.3=0
digclock.1.time.4=200
digclock.1.state.4=1
digclock.1.time.5=200
digclock.1.state.5=0
digclock.1.time.6=400
digclock.1.state.6=1

digclock.2.label=ТАКТ
digclock.2.time.1=10
digclock.2.state.1=0
digclock.2.time.2=50
digclock.2.state.2=1
digclock.2.time.3=140
digclock.2.state.3=0
digclock.2.time.4=10
digclock.2.state.4=0
digclock.2.time.5=50
digclock.2.state.5=1
digclock.2.time.6=140
digclock.2.state.6=0
digclock.2.time.7=10
digclock.2.state.7=0
digclock.2.time.8=50
digclock.2.state.8=1
digclock.2.time.9=140
digclock.2.state.9=0
digclock.2.time.10=10
digclock.2.state.10=0
digclock.2.time.11=50
digclock.2.state.11=1
digclock.2.time.12=140
digclock.2.state.12=0
digclock.2.time.13=10
digclock.2.state.13=0
digclock.2.time.14=50
digclock.2.state.14=1
digclock.2.time.15=140
digclock.2.state.15=0
digclock.2.time.16=10
digclock.2.state.16=0
digclock.2.time.17=50
digclock.2.state.17=1
digclock.2.time.18=140
digclock.2.state.18=0
digclock.2.time.19=10
digclock.2.state.19=0
digclock.2.time.20=50
digclock.2.state.20=1
digclock.2.time.21=140
digclock.2.state.21=0
digclock.2.time.22=10
digclock.2.state.22=0
digclock.2.time.23=50
digclock.2.state.23=1
digclock.2.time.24=140
digclock.2.state.24=0

#
#   Маркеры и ожидаемые значения на тестовых интервалах
#

marker.1.label=Q
marker.1.time.1=200
marker.1.state.1=0
marker.1.time.2=200
marker.1.state.2=1
marker.1.time.3=200
marker.1.state.3=0
marker.1.time.4=200
marker.1.state.4=0
marker.1.time.5=200
marker.1.state.5=1
marker.1.time.6=200
marker.1.state.6=0
marker.1.time.7=200
marker.1.state.7=1
```

```
marker.1.time.8=200
marker.1.state.8=0

marker.2.label=NOTQ
marker.2.time.1=200
marker.2.state.1=1
marker.2.time.2=200
marker.2.state.2=0
marker.2.time.3=200
marker.2.state.3=1
marker.2.time.4=200
marker.2.state.4=1
marker.2.time.5=200
marker.2.state.5=0
marker.2.time.6=200
marker.2.state.6=1
marker.2.time.7=200
marker.2.state.7=0
marker.2.time.8=200
marker.2.state.8=1
```

Приложение 3. Набор тестовых программ для оценки производительности

Тест на языке C: Life

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>

#define LiveVal 16

int wantedTime = 1000;
double *benchmarks;
double light_benchmarks[10] = {2549,2724,3011,2985,3153,3056,2442,793,219,90.5};
double heavy_benchmarks[10] = {2536,2719,2996,1430,1047,716,402,88.2,21.0,8.36};
int useBenchmark = 0;
int use_Sizes[10] = {5,6,8,10,15,25,50,250,1000,2500};
char *c_bmark_list;

int max_x, max_y;
int **board;
int cell_live_set = (1<<3) | 1<<(2+LiveVal) | 1<<(3+LiveVal);
int *rowOne,*rowZero,*topRow,*midRow,*botRow,*rowUsage;
int show_table = 0;
int lightLoad = 1;
#define rerunMainLoopCount 5
int rateFactor;
double conv_to_msecs = (double)1000.0/(double)CLOCKS_PER_SEC;

double calcRate(long cycles_per_time, long max_time);
void calcPerformance(double it_rates[], int usecomma);
clock_t getTimeAtTick();
void pretty_number(double num, char* buff);

void Life(int size);
void set_bline(int y,int row[]);
void set_line(int y,int alive[]);
void display();
void calc_new();
void do_row(int y,int top_row[],int mid_row[],int bot_row[]);
void change_board_row(int y, int score[]);
void update(int x,int top_row[],int mid_row[],int bot_row[]);
void update_sides_m1(int score_row[], int index);
void update_sides(int score_row[], int index);
void update_sides_p1(int score_row[], int index);
void update_point_three_m1(int score_row[], int index);
void update_point_three(int score_row[], int index);
void update_point_three_p1(int score_row[], int index);
void init_arrays();


int main(int argc, char **argv) {
  int c,d,try_arg = 1;
  int alength = sizeof(use_Sizes) / sizeof(int);
  double rate,cycles;
  double it_rates[rerunMainLoopCount];

  if (argc > try_arg) {
    if (strcmp(argv[try_arg],"-heavy") == 0) {
      lightLoad = 0;
      try_arg ++;
    }
    if (argc > try_arg) {
      wantedTime *= 10;
    }
  }
  c_bmark_list = (char*)calloc(100,sizeof(char));

  if (lightLoad) {
      printf("'Life' under light load\n");
      benchmarks = light_benchmarks;
  } else {
```

```
      printf("'Life' under heavy load\n");
      benchmarks = heavy_benchmarks;
  }
  if (useBenchmark) {
      printf("Results are relative to baseline\n");
  } else {
      printf("Results are in raw format\n");
  }
  printf("%s\t%s\t%s\t%s\n","Life","best","average","worst");
  for (c=0; c<alength; c++) {
      Life(use_Sizes[c]);

      rate = calcRate(1,wantedTime/100);
      cycles = rate*wantedTime/10.0;
      rate = calcRate((long)cycles,wantedTime/10);
      cycles = rate*wantedTime;
      for (d=0; d<rerunMainLoopCount; d++) {
        rate = rateFactor * calcRate((long)cycles,0);
        if (useBenchmark) {
          it_rates[d] = 100 * rate / benchmarks[c];
        } else {
          it_rates[d] = rate;
        }
      }
      calcPerformance(it_rates,(int)(c!=0));
  }
  printf("Averages: %s\n",c_bmark_list);
  exit(0);
}

double calcRate(long cycles_per_time, long max_time) {
  long i,big_counts, time_taken, now, was;
  double rate, time_taken_ms;

  big_counts = 0;
  time_taken = 0;
  was = now = getTimeAtTick();
  while (time_taken <= max_time) {
    big_counts ++;
    for (i=0; i<cycles_per_time; i++) {
      calc_new();
    }
    now = clock();
    time_taken = now-was;
  }
  time_taken_ms = time_taken * conv_to_msecs;
  rate = (double)(big_counts * cycles_per_time)/(double)time_taken_ms;
  rateFactor = max_x; /* more linear than max_x*max_y */
  return rate;
}

static int doublecompare(const void *i, const void *j) {
    if (*(double*)i > *(double*)j) return (1);
    if (*(double*)i < *(double*)j) return (-1);
    return (0);
}

void calcPerformance(double it_rates[], int usecomma) {
    int rep,j;
    double iter_rate;
    char *output, *outputs[3];
    output = (char*)calloc(200,sizeof(char));

    qsort((void*)it_rates,5,sizeof(double),doublecompare);
    for (rep=0; rep<(rerunMainLoopCount-3); rep++) {
        it_rates[1] += it_rates[rep+2];
    }
    it_rates[1] /= (rerunMainLoopCount-2);
    it_rates[2] = it_rates[rerunMainLoopCount-1];

    for (rep=0; rep<3; rep++) {
        iter_rate = it_rates[rep];
        outputs[rep] = (char*)calloc(15,sizeof(char));
        pretty_number(iter_rate,outputs[rep]);
    }
    for (j=2; j>=0; j--) {
      if (j != 2) output = strcat(output,"\t");
```

```
      output = strcat(output,outputs[j]);
    }
    printf("%d\t%s\n", max_x,output);

    if (usecomma) {
      c_bmark_list = strcat(c_bmark_list,",");
    }
    c_bmark_list = strcat(c_bmark_list,outputs[1]);

    free(output);
    for (rep=0; rep<3; rep++) free(outputs[rep]);
}

clock_t getTimeAtTick() {
  clock_t now,was;
  was = now = clock();
  while (was >= now) {
    now = clock();
  }
  return now;
}

void pretty_number(double num, char* buff) {
    int point;
    char style[] = "%.xf";
    double cutoff = 120.0;

    if (num > cutoff) {
        sprintf(buff,"%.0f",num);
        return;
    }
    if (num < 0.001) {
        sprintf(buff,"%.0f",0.0);
        return;
    }

    point = 0;
    while (num < cutoff) {
        cutoff *= 0.1;
        point ++;
    }
    style[2] = 48+point;
    sprintf(buff,style,num);
    return;
}

void Life(int size) {
  int setup1[] = {3,0,1,2};
  int setup2[] = {1,2};
  int setup3[] = {1,1};
  int block_width  = 5;
  int block_height = 5;
  int x,y;
  int *init0,*init1,*init2;

  max_x = size;
  max_y = size;
  init_arrays();

  if (lightLoad) {
    set_line(2,setup1);
    set_line(1,setup2);
    set_line(0,setup3);
    return;
  }

  init0 = (int*)calloc(max_x,sizeof(int));
  init1 = (int*)calloc(max_x,sizeof(int));
  init2 = (int*)calloc(max_x,sizeof(int));
  for (x=0; x<max_x; x++) {
    init2[x] = 0;
    init1[x] = 0;
    init0[x] = 0;
  }

  for (x=0; x<=(max_x-block_width); x+= block_width) {
    init2[x+0] = 1;
```

```
    init2[x+1] = 1;
    init2[x+2] = 1;
    init1[x+2] = 1;
    init0[x+1] = 1;
  }
  for (y=0; y<=(max_y-block_height); y+= block_height) {
    set_bline(y+2,init2);
    set_bline(y+1,init1);
    set_bline(y+0,init0);
  }
  free(init0);free(init1);free(init2);

}

void set_line(int y,int alive[]) {
  int *board_row;
  int c;
  board_row = board[y];
  for (c=0; c<=alive[0]; c++) {
    board_row[c] = alive[c];
  }
}
void set_bline(int y,int row[]) {
  int *board_row;
  int c;
  int live_count = 0;

  for (c=0; c<max_x; c++) {
    if (row[c] == 1) live_count ++;
  }

  board_row = board[y];
  board_row[0] = live_count;
  live_count = 1;
  for (c=0; c<max_x; c++) {
    if (row[c] != 1) continue;
    board_row[live_count] = c;
    live_count ++;
  }
}

void show_board_edge_line() {
  int c;
  printf("+");
  for (c=0; c<max_x; c++) {
    printf("-");
  }
  printf("+\n");
}

void display() {
  int y,c,poi,ov;
  int *board_row;
  char *output;

  output = (char*)calloc(max_x+1,sizeof(char));
  output[max_x] = 0;

  show_board_edge_line();
  for (y=max_y-1; y>=0; y--) {
    board_row = board[y];
    poi = 0;
    for (c=0; c<max_x; c++) output[c] = ' ';
    for (c=1;c<=board_row[0] ;c++) {
      ov = board_row[c];
      if (ov>max_x) {
      } else {
        output[ov] = '#';
      }
    }

    printf("|%s|\n",output);
  }
  free(output);
  show_board_edge_line();
}
```

```
void calc_new() {
  int *tmp_row;
  int y,c;
  do_row(0,rowOne,rowZero,topRow);
  do_row(max_y-1,rowZero,topRow,midRow);
  for (y=max_y-2; y>2; y--) {
    do_row(y,topRow,midRow,botRow);
    change_board_row(y+1,topRow);

    tmp_row = topRow;
    topRow = midRow;
    midRow = botRow;
    botRow = tmp_row;
    for (c=0;c<max_x;c++) botRow[c] = 0;
  }
  do_row(2,topRow,midRow,rowOne);
  change_board_row(3,topRow);
  do_row(1,midRow,rowOne,rowZero);
  change_board_row(2,midRow);
  change_board_row(1,rowOne);
  change_board_row(0,rowZero);
}

void do_row(int y,int top_row[],int mid_row[],int bot_row[]) {
  int *board_row;
  int num_live;
  int c,top,bot,x;

  board_row = board[y];
  num_live = board_row[0];
  for (c=1;c<=num_live ;c++) {
    x = board_row[c];
    update(x,top_row,mid_row,bot_row);
  }
  if (num_live==0) return;

  top = y+1;
  bot = y-1;
  if (top==max_y) top=0;
  if (bot==-1) bot=max_y-1;
  rowUsage[y]   += num_live;
  rowUsage[top] += num_live;
  rowUsage[bot] += num_live;
}

void change_board_row(int y, int score[]) {
  int insert_pos;
  int *board_row;
  int usage;
  int c;

  insert_pos = 1;
  board_row = board[y];
  usage = rowUsage[y];

  rowUsage[y] = 0;
  if (usage<3) {
    if (usage>0) {
      board_row[0] = 0;
      for (c=0;c<max_x;c++) score[c] = 0;
    }
    return;
  }
  for (c=0; c<max_x; c++) {
    if (((1<<score[c]) & cell_live_set)>0) {
      board_row[insert_pos] = c;
      insert_pos ++;
    }
    score[c] = 0;
  }
  board_row[0] = insert_pos-1;
}

void update(int x,int top_row[],int mid_row[],int bot_row[]) {
  if (x==0) {
    update_point_three_m1(top_row,x);
    update_sides_m1(mid_row,x);
```

```
    update_point_three_m1(bot_row,x);
    return;
  }
  if (x==(max_x-1)) {
    update_point_three_p1(top_row,x);
    update_sides_p1(mid_row,x);
    update_point_three_p1(bot_row,x);
    return;
  }
  update_point_three(top_row,x);
  update_sides(mid_row,x);
  update_point_three(bot_row,x);
}

void update_sides_m1(int score_row[], int index) {
  score_row[max_x-1] += 1;
  score_row[index-0] += LiveVal;
  score_row[index+1] += 1;
  return;
}
void update_sides(int score_row[], int index) {
  score_row[index-1] += 1;
  score_row[index-0] += LiveVal;
  score_row[index+1] += 1;
  return;
}
void update_sides_p1(int score_row[], int index) {
  score_row[index-1] += 1;
  score_row[index-0] += LiveVal;
  score_row[0      ] += 1;
  return;
}

void update_point_three_m1(int score_row[], int index) {
  score_row[max_x-1] += 1;
  score_row[index-0] += 1;
  score_row[index+1] += 1;
  return;
}
void update_point_three(int score_row[], int index) {
  score_row[index-1] += 1;
  score_row[index-0] += 1;
  score_row[index+1] += 1;
  return;
}
void update_point_three_p1(int score_row[], int index) {
  score_row[index-1] += 1;
  score_row[index-0] += 1;
  score_row[0      ] += 1;
  return;
}

void init_arrays() {
  int c;
  if (board) {
    free(board);
    free(rowOne);
    free(rowZero);
    free(topRow);
    free(midRow);
    free(botRow);
    free(rowUsage);
  }

  board = (int**) calloc(max_y,sizeof(int*));
  for (c=0;c<max_y;c++) {
    board[c] = (int*)calloc(max_x,sizeof(int));
    board[c][0] = 0;
  }

  rowOne  = (int*)calloc(max_x,sizeof(int));
  rowZero = (int*)calloc(max_x,sizeof(int));
  topRow  = (int*)calloc(max_x,sizeof(int));
  midRow  = (int*)calloc(max_x,sizeof(int));
  botRow  = (int*)calloc(max_x,sizeof(int));

  rowUsage = (int*)calloc(max_y,sizeof(int));
```

```
  return;
}
```

Тест на языке C: Life (infinite)

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <limits.h>

#define LiveVal 16

int wantedTime = 1000;
double benchmarks[10] = {234,251,255,257,260,249,232,221,209,199};
int useBenchmark = 0;
int use_Sizes[10] = {1,2,3,4,5,10,15,20,25,30};
char *c_bmark_list;

int MAX_X = INT_MAX;
int MAX_Y = INT_MAX;
int MIN_X = INT_MIN;
int MIN_Y = INT_MIN;
int PerC = 2;
int *board;
int board_size,board_length;
int cell_live_set = (1<<3) | 1<<(2+LiveVal) | 1<<(3+LiveVal);
int show_table = 0;
#define rerunMainLoopCount 5
int rateFactor;
double conv_to_msecs = (double)1000.0/(double)CLOCKS_PER_SEC;

double calcRate(long cycles_per_time, long max_time);
void calcPerformance(double it_rates[], int usecomma);
clock_t getTimeAtTick();
void pretty_number(double num, char* buff);

void infilife(int size);
void display(int min_x, int min_y, int max_x, int max_y);
void calc_new();
int addVote(int **score,int x, int y, int val);

int main(int argc, char **argv) {
  int c,d;
  int alength = sizeof(use_Sizes) / sizeof(int);
  double rate,cycles;
  double it_rates[rerunMainLoopCount];

  if (argc > 1) wantedTime *= 10;
  c_bmark_list = (char*)calloc(100,sizeof(char));

  if (useBenchmark) {
      printf("Results are relative to baseline\n");
  } else {
      printf("Results are in raw format\n");
  }
  printf("%s\t%s\t%s\t%s\n","Life","best","average","worst");
  for (c=0; c<alength; c++) {
      infilife(use_Sizes[c]);

      rate = calcRate(1,wantedTime/100);
      cycles = rate*wantedTime/10.0;
      rate = calcRate((long)cycles,wantedTime/10);
      cycles = rate*wantedTime;
      for (d=0; d<rerunMainLoopCount; d++) {
        rate = rateFactor * calcRate((long)cycles,0);
        if (useBenchmark) {
          it_rates[d] = 100 * rate / benchmarks[c];
        } else {
          it_rates[d] = rate;
        }
      }
      calcPerformance(it_rates,(int)(c!=0));
  }
  printf("Averages: %s\n",c_bmark_list);
  exit(0);
}
```

```
double calcRate(long cycles_per_time, long max_time) {
  long i,big_counts, time_taken, now, was;
  double rate, time_taken_ms;

  big_counts = 0;
  time_taken = 0;
  was = now = getTimeAtTick();
  while (time_taken <= max_time) {
    big_counts ++;
    for (i=0; i<cycles_per_time; i++) {
      calc_new();
    }
    now = clock();
    time_taken = now-was;
  }
  time_taken_ms = time_taken * conv_to_msecs;
  rate = (double)(big_counts * cycles_per_time)/(double)time_taken_ms;
  rateFactor = board_length;
  return rate;
}

static int doublecompare(const void *i, const void *j) {
    if (*(double*)i > *(double*)j) return (1);
    if (*(double*)i < *(double*)j) return (-1);
    return (0);
}

void calcPerformance(double it_rates[], int usecomma) {
    int rep,j;
    double iter_rate;
    char *output, *outputs[3];
    output = (char*)calloc(200,sizeof(char));

    qsort((void*)it_rates,5,sizeof(double),doublecompare);
    for (rep=0; rep<(rerunMainLoopCount-3); rep++) {
        it_rates[1] += it_rates[rep+2];
    }
    it_rates[1] /= (rerunMainLoopCount-2);
    it_rates[2] = it_rates[rerunMainLoopCount-1];

    for (rep=0; rep<3; rep++) {
        iter_rate = it_rates[rep];
        outputs[rep] = (char*)calloc(15,sizeof(char));
        pretty_number(iter_rate,outputs[rep]);
    }
    for (j=2; j>=0; j--) {
      if (j != 2) output = strcat(output,"\t");
      output = strcat(output,outputs[j]);
    }
    printf("%d\t%s\n", board_size*5,output);

    if (usecomma) {
      c_bmark_list = strcat(c_bmark_list,",");
    }
    c_bmark_list = strcat(c_bmark_list,outputs[1]);

    free(output);
    for (rep=0; rep<3; rep++) free(outputs[rep]);
}

clock_t getTimeAtTick() {
  clock_t now,was;
  was = now = clock();
  while (was >= now) {
    now = clock();
  }
  return now;
}

void pretty_number(double num, char* buff) {
    int point;
    char style[] = "%.xf";
    double cutoff = 120.0;

    if (num > cutoff) {
        sprintf(buff,"%.0f",num);
```

```
        return;
    }
    if (num < 0.001) {
        sprintf(buff,"%.0f",0.0);
        return;
    }

    point = 0;
    while (num < cutoff) {
        cutoff *= 0.1;
        point ++;
    }
    style[2] = 48+point;
    sprintf(buff,style,num);
    return;
}

void infilife(int size) {
    int alive_per_group = 5; // per glider
    int set,col,row,x,y;
    int x_gap = 5;
    int y_gap = 5;
    int end = size*size * alive_per_group * PerC;
    if (board) free(board);
    board = (int*)calloc(end,sizeof(int));
    board_size = size;
    board_length = end;

    set = 0;
    for (col=0; col<size; col++) {
        y = col*y_gap;
        for (row=0; row<size; row++) {
            x = row*x_gap;
            board[set+0] = x+0; board[set+1] = y+2;
            board[set+2] = x+1; board[set+3] = y+2;
            board[set+4] = x+2; board[set+5] = y+2;
            board[set+6] = x+2; board[set+7] = y+1;
            board[set+8] = x+1; board[set+9] = y+0;
            set += alive_per_group * PerC;
        }
    }
}

void display(int min_x, int min_y, int max_x, int max_y) {
  int y,x,i,ypos;
  int width = max_y-min_y;
  int height = max_x-min_x;
  char *alive;

  alive = (char*)calloc(width*height,sizeof(char));

  for (i=0; i<(width*height); i++) {
    alive[i] = ' ';
  }

  for (i=0; i<board_length; i += PerC) {
    x = board[i + 0];
    y = board[i + 1];
    if (x <  min_x) continue;
    if (y <  min_y) continue;
    if (x >= max_x) continue;
    if (y >= max_y) continue;
    ypos = (y-min_y) * width;
    alive[ypos + (x-min_x)] = '#';
  }

  printf("\t+");
  for (x=0; x<width; x++) printf("-");
  printf("+\n");
  for (y=width*(height-1); y>=0; y -= width) {
    printf("%d\t|",y/width);
    for (x=0;x<width;x++) printf("%c",alive[y+x]);
    printf("|\n");
  }

  printf("\t+");
  for (x=0; x<width; x++) printf("-");
```

```
  printf("+\n\n");
  free(alive);
}

void calc_new() {
    int i,set_length,xpos_c,ypos_c,xpos_l,xpos_r,ypos_b,ypos_t;
    int poi,*new_board,*set,seti,x,y,s;
    int alive_next = 0;
    int **score;
    int alive = board_length / 2;

    score = (int**)calloc(alive,sizeof(int*));

    for (i=0; i<board_length; i += PerC) {
        xpos_c = board[i + 0];
        ypos_c = board[i + 1];

        xpos_l = (xpos_c < MAX_X)? xpos_c-1: MIN_X;
        xpos_r = (xpos_c > MIN_X)? xpos_c+1: MAX_X;
        ypos_b = (ypos_c < MAX_Y)? ypos_c-1: MIN_Y;
        ypos_t = (ypos_c > MIN_Y)? ypos_c+1: MAX_Y;

        alive_next += addVote(score,xpos_l,ypos_t, 1);
        alive_next += addVote(score,xpos_c,ypos_t, 1);
        alive_next += addVote(score,xpos_r,ypos_t, 1);
        alive_next += addVote(score,xpos_l,ypos_c, 1);
        alive_next += addVote(score,xpos_c,ypos_c, LiveVal);
        alive_next += addVote(score,xpos_r,ypos_c, 1);
        alive_next += addVote(score,xpos_l,ypos_b, 1);
        alive_next += addVote(score,xpos_c,ypos_b, 1);
        alive_next += addVote(score,xpos_r,ypos_b, 1);
    }

    poi = 0;
    new_board = (int*)malloc(alive_next * PerC * sizeof(int));
    for (seti=0; seti<alive; seti++) {
        set = score[seti];
        if (set == 0) continue;
        set_length = set[0];
        for (i=1; i<set_length; i+=3) {
            x = set[i+0];
            y = set[i+1];
            s = set[i+2];
            if (((1 << s) & cell_live_set) <= 0) continue;
            new_board[poi + 0] = x;
            new_board[poi + 1] = y;
            poi += PerC;
        }
    }

    free(board);
    board = new_board;
    for (i=0; i<alive; i++)
        if (score[i]) free(score[i]);
    free(score);
}

int addVote(int **score,int x, int y, int val) {
    int *set,i,hash,set_length,votes,was_alive,now_alive,*nset;
    int score_length = board_length / 2;
    //    printf("SL: %d\n",score_length);
    hash = (x+y) ^ ((x<<2) - (y>>1));
    if (hash<0) hash = -hash;
    hash = hash % score_length;
    set = score[hash];
    if (set == 0) {
        set = (int*)malloc(4*sizeof(int));
        set[0] = 4;
        set[1] = x;
        set[2] = y;
        set[3] = val;
        score[hash] = set;
        return 0;
    }

    set_length = set[0];
    //    printf("SL: %d\n",set_length);
```

```
    for (i=1; i<set_length; i+=3) {
        if ((set[i+0] != x) || (set[i+1] != y)) continue;
        votes = set[i+2];
        was_alive = 0;
        if (((1 << votes) & cell_live_set) > 0) was_alive = 1;
        votes += val;
        now_alive = 0;
        if (((1 << votes) & cell_live_set) > 0) now_alive = 1;
        set[i+2] = votes;
        return (now_alive - was_alive);
    }

    nset = (int*)malloc((set_length+3) * sizeof(int));
    nset[0] = set_length + 3;
    for (i=1; i<set_length; i++) {
        nset[i] = set[i];
    }
    free(set);
    nset[set_length + 0] = x;
    nset[set_length + 1] = y;
    nset[set_length + 2] = val;
    score[hash] = nset;
    return 0;
}
```

Тест на языке C: Fibonacci

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>

int wantedTime = 1000;
double benchmarks[8] = {16837,15561,14976,14843,14807,14803,14805,14801};
int useBenchmark = 0;
int fibNums[8] = {5,7,10,12,15,20,22,25};
char *c_bmark_list;

int fib_num, rateFactor;
double conv_to_msecs = (double)1000.0/(double)CLOCKS_PER_SEC;
#define rerunMainLoopCount 5

int doFib(int fib);
clock_t getTimeAtTick();
double calcRate(int fib_num, long cycles_per_time, long max_time);
void calcPerformance(double it_rates[], int usecomma);
void pretty_number(double num, char* buff);

int main(int argc, char **argv) {
  int c,d;
  int alength = sizeof(fibNums) / sizeof(int);
  double rate,cycles;
  double it_rates[rerunMainLoopCount];

  if (argc > 1) wantedTime *= 10;
  c_bmark_list = (char*)calloc(100,sizeof(char));

  if (useBenchmark) {
      printf("Results are relative to baseline\n");
  } else {
      printf("Results are in raw format\n");
  }
  printf("%s\t%s\t%s\t%s\n","FibNum","best","average","worst");
  for (c=0; c<alength; c++) {
    fib_num = fibNums[c];
    rate = calcRate(fib_num,1,wantedTime/100);
    cycles = rate*wantedTime/10.0;
    rate = calcRate(fib_num,(long)cycles,wantedTime/10);
    cycles = rate*wantedTime;
    for (d=0; d<rerunMainLoopCount; d++) {
      rate = rateFactor * calcRate(fib_num,(long)cycles,0);
      if (useBenchmark) {
        it_rates[d] = 100 * rate / benchmarks[c];
      } else {
        it_rates[d] = rate;
      }
    }
```

```
    calcPerformance(it_rates,(int)(c!=0));
  }

  printf("Averages: %s\n",c_bmark_list);
  exit(0);
}

int doFib(int fib) {
  if (fib < 2) return 1;
  return doFib(fib-1) + doFib(fib-2);
}

double calcRate(int fib_num, long cycles_per_time, long max_time) {
  long i,big_counts, time_taken, now, was;
  double rate, time_taken_ms;

  big_counts = 0;
  time_taken = 0;
  was = now = getTimeAtTick();
  while (time_taken <= max_time) {
    big_counts ++;
    for (i=0; i<cycles_per_time; i++) {
       rateFactor = doFib(fib_num);
    }
    now = clock();
    time_taken = now-was;
  }
  time_taken_ms = time_taken * conv_to_msecs;
  rate = (double)(big_counts * cycles_per_time)/(double)time_taken_ms;
  return rate;
}

clock_t getTimeAtTick() {
  clock_t now,was;
  was = now = clock();
  while (was >= now) {
    now = clock();
  }
  return now;
}
static int doublecompare(const void *i, const void *j) {
    if (*(double*)i > *(double*)j) return (1);
    if (*(double*)i < *(double*)j) return (-1);
    return (0);
}

void calcPerformance(double it_rates[], int usecomma) {
    int rep,j;
    double iter_rate;
    char *output, *outputs[3];
    output = (char*)calloc(200,sizeof(char));

    qsort((void*)it_rates,5,sizeof(double),doublecompare);
    for (rep=0; rep<(rerunMainLoopCount-3); rep++) {
       it_rates[1] += it_rates[rep+2];
    }
    it_rates[1] /= (rerunMainLoopCount-2);
    it_rates[2] = it_rates[rerunMainLoopCount-1];

    for (rep=0; rep<3; rep++) {
       iter_rate = it_rates[rep];
       outputs[rep] = (char*)calloc(15,sizeof(char));
       pretty_number(iter_rate,outputs[rep]);
    }
    for (j=2; j>=0; j--) {
      if (j != 2) output = strcat(output,"\t");
      output = strcat(output,outputs[j]);
    }
    printf("%d\t%s\n", fib_num,output);

    if (usecomma) {
      c_bmark_list = strcat(c_bmark_list,",");
    }
    c_bmark_list = strcat(c_bmark_list,outputs[1]);

    free(output);
    for (rep=0; rep<3; rep++) free(outputs[rep]);
```

```
}

void pretty_number(double num, char* buff) {
    int point;
    char style[] = "%.xf";
    double cutoff = 120.0;

    if (num > cutoff) {
        sprintf(buff,"%.0f",num);
        return;
    }
    if (num < 0.001) {
        sprintf(buff,"%.0f",0.0);
        return;
    }

    point = 0;
    while (num < cutoff) {
        cutoff *= 0.1;
        point ++;
    }
    style[2] = 48+point;
    sprintf(buff,style,num);
    return;
}
```

Тест на языке C: FFT

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <math.h>

int wantedTime = 1000;
double benchmarks[11] = {775,991,1158,1260,1332,1219,957,273,232,183,134};
int useBenchmark = 0;
int use_Sizes[11] = {3,4,5,6,8,10,12,13,14,15,16};
char *c_bmark_list;

#define rerunMainLoopCount 5
#ifndef M_PI
#define M_PI 3.141593
#endif

int rateFactor;
double conv_to_msecs = (double)1000.0/(double)CLOCKS_PER_SEC;
double *testArray;
double JDKtotal = 0.0;
double JDKtotali = 0.0;

double* makeRandom(int n);
double test(double data[]);
void FFT(int size);
int mylog2(int n);
void bitreverse(double data[]);
void transform_internal (double data[], int direction);

double calcRate(long cycles_per_time, long max_time);
void calcPerformance(double it_rates[], int usecomma);
clock_t getTimeAtTick();
void pretty_number(double num, char* buff);


int main(int argc, char **argv) {
  int c,d;
  int alength = sizeof(use_Sizes) / sizeof(int);
  double rate,cycles;
  double it_rates[rerunMainLoopCount];

  if (argc > 1) wantedTime *= 10;
  c_bmark_list = (char*)calloc(100,sizeof(char));

  if (useBenchmark) {
      printf("Results are relative to baseline\n");
  } else {
      printf("Results are in raw format\n");
```

```
  }

  printf("%s\t%s\t%s\t%s\n","FFT","best","average","worst");
  for (c=0; c<alength; c++) {
      rateFactor = 1 << use_Sizes[c];
      FFT(rateFactor);

      rate = calcRate(1,wantedTime/100);
      cycles = rate*wantedTime/10.0;
      rate = calcRate((long)cycles,wantedTime/10);
      cycles = rate*wantedTime;
      for (d=0; d<rerunMainLoopCount; d++) {
        rate = rateFactor * calcRate((long)cycles,0);
        if (useBenchmark) {
          it_rates[d] = 100 * rate / benchmarks[c];
        } else {
          it_rates[d] = rate;
        }
      }
      printf("%d\t", use_Sizes[c]);
      calcPerformance(it_rates,(int)(c!=0));
  }

  printf("Averages: %s\n",c_bmark_list);
  exit(0);
}

double calcRate(long cycles_per_time, long max_time) {
  long i,big_counts, time_taken, now, was;
  double rate, time_taken_ms;
  double diff = 0.0;

  big_counts = 0;
  time_taken = 0;
  was = now = getTimeAtTick();
  while (time_taken <= max_time) {
    big_counts ++;
    for (i=0; i<cycles_per_time; i++) {
        diff += test(testArray);
    }
    now = clock();
    time_taken = now-was;
  }
  time_taken_ms = time_taken * conv_to_msecs;
  rate = (double)(big_counts * cycles_per_time)/(double)time_taken_ms;
  if (diff > 10) {
      fprintf(stderr,"Diff too big? %f\n",diff);
  }
  return rate;
}

static int doublecompare(const void *i, const void *j) {
    if (*(double*)i > *(double*)j) return (1);
    if (*(double*)i < *(double*)j) return (-1);
    return (0);
}

void calcPerformance(double it_rates[], int usecomma) {
    int rep,j;
    double iter_rate;
    char *output, *outputs[3];
    output = (char*)calloc(200,sizeof(char));

    qsort((void*)it_rates,5,sizeof(double),doublecompare);
    for (rep=0; rep<(rerunMainLoopCount-3); rep++) {
        it_rates[1] += it_rates[rep+2];
    }
    it_rates[1] /= (rerunMainLoopCount-2);
    it_rates[2] = it_rates[rerunMainLoopCount-1];

    for (rep=0; rep<3; rep++) {
        iter_rate = it_rates[rep];
        outputs[rep] = (char*)calloc(15,sizeof(char));
        pretty_number(iter_rate,outputs[rep]);
    }
    for (j=2; j>=0; j--) {
      if (j != 2) output = strcat(output,"\t");
```

```
      output = strcat(output,outputs[j]);
    }
    printf("%s\n",output);

    if (usecomma) {
      c_bmark_list = strcat(c_bmark_list,",");
    }
    c_bmark_list = strcat(c_bmark_list,outputs[1]);

    free(output);
    for (rep=0; rep<3; rep++) free(outputs[rep]);
}

clock_t getTimeAtTick() {
  clock_t now,was;
  was = now = clock();
  while (was >= now) {
    now = clock();
  }
  return now;
}

void pretty_number(double num, char* buff) {
    int point;
    char style[] = "%.xf";
    double cutoff = 120.0;

    if (num > cutoff) {
        sprintf(buff,"%.0f",num);
        return;
    }
    if (num < 0.001) {
        sprintf(buff,"%.0f",0.0);
        return;
    }

    point = 0;
    while (num < cutoff) {
        cutoff *= 0.1;
        point ++;
    }
    style[2] = 48+point;
    sprintf(buff,style,num);
    return;
}

  /** Compute Fast Fourier Transform of (complex) data, in place.*/
void transform (double data[]) {
  int nd= rateFactor*2;
  int i;

  transform_internal(data, -1);

  for (i=0; i<nd; i++) {
    JDKtotal += data[i];
  }
}

/** Compute Inverse Fast Fourier Transform of (complex) data, in place.*/
void inverse (double data[]) {
  int nd,n,i;
  double norm;

  transform_internal(data, +1);
  nd=rateFactor*2;
  n =nd/2;
  norm=1/((double) n);
  for (i=0; i<nd; i++)
      data[i] *= norm;

  for (i=0; i<nd; i++) {
    JDKtotali += data[i];
  }

}

void FFT(int size) {
```

```
  if (testArray) free(testArray);
  testArray = makeRandom(size);
}

double test(double data[]){
  int nd,i;
    double *copy,diff,d;

    nd = rateFactor*2;
    copy = (double*)calloc(nd,sizeof(double));
    for (i=0; i<nd; i++)
      copy[i] = data[i];

    transform(data);
    inverse(data);
    diff = 0.0;
    for (i=0; i<nd; i++) {
      d = data[i]-copy[i];
      diff += d*d;
    }

    free(copy);
    return sqrt(diff/nd);
}

double* makeRandom(int n) {
  double *data;
  int i;
  int nd = 2*n;

  data = (double*)calloc(nd,sizeof(double));
  srand(1231);
  for (i=0; i<nd; i++) {
    /* Okay, this isn't exactly random, but it's simple and cross-platform, though the actual
numbers don't make a lot of difference. */
    data[i]= (double)((i*i+1)%10) / 10.0;
  }
  return data;
}

int mylog2(int n) {
  int k, log = 0;
  for(k=1; k < n; k *= 2, log++);
  if (n != (1 << log)) {
    /*      fprintf(stderr,"FFT: Data length is not a power of 2!: %d\n",n);
           exit(0);*/
  }
  return log;
}

void transform_internal(double data[], int direction) {
  int n= rateFactor;
  int logn, bit, dual,b,a,i,j;
  double w_real,w_imag,theta,s,s2,t,wd_real,wd_imag,tmp_real,tmp_imag,z1_real,z1_imag;

  if (n == 1) return;
  logn = mylog2(n);

  /* bit reverse the input data for decimation in time algorithm */
  bitreverse(data) ;

  /* apply fft recursion */
  for (bit = 0, dual = 1; bit < logn; bit++, dual *= 2) {
    w_real = 1.0;
    w_imag = 0.0;

    theta = 2.0 * direction * M_PI / (2.0 * (double) dual);
    s = sin(theta);
    t = sin(theta / 2.0);
    s2 = 2.0 * t * t;

    /* a = 0 */
    for (b = 0; b < n; b += 2 * dual) {
      i = 2*b ;
      j = 2*(b + dual);

      wd_real = data[j] ;
```

```
      wd_imag = data[j+1] ;

      data[j]   = data[i]   - wd_real;
      data[j+1] = data[i+1] - wd_imag;
      data[i]  += wd_real;
      data[i+1]+= wd_imag;
      }

    for (a = 1; a < dual; a++) {
      {
       tmp_real = w_real - s * w_imag - s2 * w_real;
       tmp_imag = w_imag + s * w_real - s2 * w_imag;
       w_real = tmp_real;
       w_imag = tmp_imag;
      }
      for (b = 0; b < n; b += 2 * dual) {
       i = 2*(b + a);
       j = 2*(b + a + dual);

       z1_real = data[j];
       z1_imag = data[j+1];

       wd_real = w_real * z1_real - w_imag * z1_imag;
       wd_imag = w_real * z1_imag + w_imag * z1_real;

       data[j]   = data[i]   - wd_real;
       data[j+1] = data[i+1] - wd_imag;
       data[i]  += wd_real;
       data[i+1]+= wd_imag;
      }
    }
  }
}


void bitreverse(double data[]) {
  int n= rateFactor;
  int i,j,ii,jj,k;
  double tmp_real,tmp_imag;

  for (i = 0, j=0; i < n - 1; i++) {
    ii = 2*i;
    jj = 2*j;
    k = n / 2 ;
    if (i < j) {
      tmp_real    = data[ii];
      tmp_imag    = data[ii+1];
      data[ii]   = data[jj];
      data[ii+1] = data[jj+1];
      data[jj]   = tmp_real;
      data[jj+1] = tmp_imag;
    }

    while (k <= j) {
      j = j - k ;
      k = k / 2 ;
    }
    j += k ;
  }
}
```

Тест на языке Java: Life

```
import java.util.*;

public class life_light {
    static long wantedTime = 1000;
    static double[] light_benchmarks = new double[]
{2549,2724,3011,2985,3153,3056,2442,793,219,90.5}; // compared to C results
    static double[] heavy_benchmarks = new double[]
{2536,2719,2996,1430,1047,716,402,88.2,21.0,8.36}; // compared to C results
    static double[] benchmarks;
    static boolean useBenchmark = false;

    static String java_bmark_list = "";
    static long rateFactor;
    static int rerunMainLoopCount = 5; // has to be at least 3
```

```
static boolean show_table = false;
static boolean lightLoad = true;
static int[] use_Sizes = new int[] {5,6,8,10,15,25,50,250,1000,2500};

int max_x, max_y;
int[][] board;
final int LiveVal = 16;
int cell_live_set = (1<<3) | 1<<(3+LiveVal) | 1<<(2+LiveVal);
int[] rowOne,rowZero,topRow,midRow,botRow,rowUsage;

public static void main(String[] args) {
    int try_arg = 0;
    if (args.length > try_arg) {
        if (args[try_arg].equals("-heavy")) {
            lightLoad = false;
            try_arg ++;
        }
        if (args.length > try_arg) {
            wantedTime *= 10;
        }
    }

    double[] it_rates = new double[rerunMainLoopCount];

    if (lightLoad) {
        System.out.println("'Life' under light load");
        benchmarks = light_benchmarks;
    } else {
        System.out.println("'Life' under heavy load");
        benchmarks = heavy_benchmarks;
    }
    if (useBenchmark) {
        System.out.println("Results are relative to baseline");
    } else {
        System.out.println("Results are in raw format");
    }
    System.out.println("Size\tbest\taverage\tworst");

    for (int c=0; c<use_Sizes.length; c++) {
        life_light life_ob = new life_light(use_Sizes[c]);
        double rate = life_ob.calcRate(1,20);
        long cycles = (long)(rate*wantedTime/10d);
        rate = life_ob.calcRate(cycles,200);
        cycles = (long)(rate*wantedTime);
        for (int d=0; d<5; d++) {
            rate = life_ob.calcRate(cycles,(long)(wantedTime*0.9));
            long check = (long)(rate*wantedTime);
            if (check > (cycles * 2)) {
                System.out.println("Adjusting "+cycles+" to "+check);
                cycles = check;
            }

            rate *= rateFactor;
            if (useBenchmark) {
                it_rates[d] = 100d * rate / benchmarks[c];
            } else {
                it_rates[d] = rate;
            }
        }
        life_ob.calcPerformance(it_rates,(c==0));
    }
    System.out.println("Averages: "+java_bmark_list);
}

double calcRate(long cycles_per_time, long max_time) {
    long big_counts = 0;
    long time_taken = 0;
    long now = getTimeAtTick();
    long was = now;
    while (time_taken <= max_time) {
        big_counts ++;
        for (long i=0; i<cycles_per_time; i++) {
            calc_new();
        }
        now = System.currentTimeMillis();
        time_taken = now-was;
    }
```

```
    double rate = (double)(big_counts * cycles_per_time)/(double)time_taken;
    rateFactor = max_x;
    return rate;
}

static long getTimeAtTick() {
    long now = System.currentTimeMillis();
    final long was = now;
    while (was >= now) {
        now = System.currentTimeMillis();
    }
    return now;
}

private void calcPerformance(double[] it_rates, boolean is_first) {
    sortArray(it_rates);
    for (int rep=0; rep<(rerunMainLoopCount-3); rep++) {
        it_rates[1] += it_rates[rep+2];
    }
    it_rates[1] /= (rerunMainLoopCount-2);
    it_rates[2] = it_rates[rerunMainLoopCount-1];

    String[] outputs = new String[3];
    for (int rep=0; rep<3; rep++) {
        outputs[rep] = pretty_number(it_rates[rep]);
    }
    String output = "";
    for (int j=2; j>=0; j--) {
        if (j != 2) output += "\t";
        output += outputs[j];
    }
    if (!is_first) java_bmark_list += ",";
    java_bmark_list += outputs[1];
    System.out.println(max_x+"\t"+output);
}

static private void sortArray(double[] array) {
    if (array.length < 2) return;
    boolean changed = true;
    while (changed) {
        changed = false;
        for (int c=1; c<array.length; c++) {
            if (array[c-1] > array[c]) {
                changed = true;
                double tmp = array[c];
                array[c] = array[c-1];
                array[c-1] = tmp;
            }
        }
    }
}

static private String pretty_number(double num) {
    double cutoff = 120;
    if (num > cutoff) return ""+(int)num;
    if (num < 0.001) return "0";
    double fac = 1d;
    while (num < cutoff) {
        num *= 10d;
        fac *= 10d;
    }
    int tt = (int)num;
    num = tt / fac;
    return ""+num;
}

public life_light(int size) {
    max_x = size;
    max_y = size;
    init_arrays();

    if (lightLoad) {
        set_line(2,new int[] {0,1,2});
        set_line(1,new int[] {2});
        set_line(0,new int[] {1});
        return;
    }
```

```
    int block_width  = 5;
    int block_height = 5;
    boolean init[][] = new boolean[3][max_x];
    for (int x=0; x<=(max_x-block_width); x+= block_width) {
        init[2][x+0] = init[2][x+1] = init[2][x+2] = true;
                                      init[1][x+2] = true;
                       init[0][x+1]                = true;
    }
    for (int y=0; y<=(max_y-block_height); y+= block_height) {
        set_line(y+2,init[2]);
        set_line(y+1,init[1]);
        set_line(y+0,init[0]);
    }
}

public void set_line(int y,int[] alive) {
    int[] board_row = board[y];
    for (int c=1; c<=alive.length; c++) {
        board_row[c] = alive[c-1];
    }
    board_row[0] = alive.length;
}
public void set_line(int y,boolean[] row) {
    int[] board_row = board[y];
    int live_count = 0;
    for (int c=0; c<row.length; c++) {
        if (row[c]) live_count ++;
    }

    board_row[0] = live_count;
    live_count = 1;
    for (int c=0; c<row.length; c++) {
        if (!row[c]) continue;
        board_row[live_count] = c;
        live_count ++;
    }
}

public void display() {
    StringBuffer output = new StringBuffer();
    boolean fl;
    for (int y=max_y; y>=-1; y--) {
        if ((y==max_y) || (y==-1)) fl=true; else fl=false;
        if (fl) output.append("\t+"); else output.append(y+"\t|");
        int r_base = output.length();
        for (int c=0; c<max_x; c++)
            if (fl) output.append("-"); else output.append(" ");
        if (fl) output.append("+\n"); else output.append("|\n");
        if (fl) continue;

        int[] board_row = board[y];

        for (int c=1;c<=board_row[0] ;c++) {
            int ov = r_base + board_row[c];
            if (ov>output.length()) {
                System.out.println("Bad "+y+" "+c+" "+ov);
            } else {
                output.setCharAt(ov,'#');
            }
        }
    }
    System.out.println(output);
}

public void calc_new() {
    int[] tmp_row;

    do_row(0,rowOne,rowZero,topRow);
    do_row(max_y-1,rowZero,topRow,midRow);
    for (int y=max_y-2; y>2; y--) {
        do_row(y,topRow,midRow,botRow);
        change_board_row(y+1,topRow);

        tmp_row = topRow;
        topRow = midRow;
        midRow = botRow;
```

```
            botRow = tmp_row;
            for (int c=0;c<botRow.length;c++) botRow[c] = 0;
        }
        do_row(2,topRow,midRow,rowOne);
        change_board_row(3,topRow);
        do_row(1,midRow,rowOne,rowZero);
        change_board_row(2,midRow);
        change_board_row(1,rowOne);
        change_board_row(0,rowZero);
    }

    private void do_row(int y,int[] top_row,int[] mid_row,int[] bot_row) {
        int[] board_row = board[y];
        int num_live = board_row[0];
        for (int c=1;c<=num_live ;c++) {
            int x = board_row[c];
            update(x,top_row,mid_row,bot_row);
        }
        if (num_live==0) return;

        int top = y+1;
        int bot = y-1;
        if (top==max_y) top=0;
        if (bot==-1) bot=max_y-1;
        rowUsage[y]   += num_live;
        rowUsage[top] += num_live;
        rowUsage[bot] += num_live;
    }

    private void change_board_row(int y, int[] score) {
        int[] board_row = board[y];
        int usage = rowUsage[y];
        rowUsage[y] = 0;
        if (usage<3) {
            //System.out.println();
            if (usage>0) {
               board_row[0] = 0;
               for (int c=0;c<score.length;c++) score[c] = 0;
            }
            return;
        }
        //System.out.print("Upd "+y+","+usage+"\t:");
        int insert_pos = 1;
        for (int c=0; c<score.length; c++) {
            if (((1<<score[c]) & cell_live_set)>0) {
               //System.out.print("["+score[c]+"]");
               board_row[insert_pos] = c;
               insert_pos ++;
            } else {
               //System.out.print("<"+score[c]+">");
            }
            score[c] = 0;
        }
        //System.out.println();
        board_row[0] = insert_pos-1;
    }

    private void update(int x,int[] top_row,int[] mid_row,int[] bot_row) {
        if (x==0) {
            update_point_three_m1(top_row,x);
            update_sides_m1(mid_row,x);
            update_point_three_m1(bot_row,x);
            return;
        }
        if (x==(max_x-1)) {
            update_point_three_p1(top_row,x);
            update_sides_p1(mid_row,x);
            update_point_three_p1(bot_row,x);
            return;
        }
        update_point_three(top_row,x);
        update_sides(mid_row,x);
        update_point_three(bot_row,x);
    }


    private void update_sides_m1(int[] score_row, int index) {
```

```
        score_row[max_x-1] += 1;
        score_row[index-0] += LiveVal;
        score_row[index+1] += 1;
    }
    private void update_sides(int[] score_row, int index) {
        score_row[index-1] += 1;
        score_row[index-0] += LiveVal;
        score_row[index+1] += 1;
    }
    private void update_sides_p1(int[] score_row, int index) {
        score_row[index-1] += 1;
        score_row[index-0] += LiveVal;
        score_row[0      ] += 1;
    }

    private void update_point_three_m1(int[] score_row, int index) {
        score_row[max_x-1] += 1;
        score_row[index-0] += 1;
        score_row[index+1] += 1;
    }
    private void update_point_three(int[] score_row, int index) {
        score_row[index-1] += 1;
        score_row[index-0] += 1;
        score_row[index+1] += 1;
    }
    private void update_point_three_p1(int[] score_row, int index) {
        score_row[index-1] += 1;
        score_row[index-0] += 1;
        score_row[0      ] += 1;
    }

    private void init_arrays() {
        board = new int[max_y][];
        for (int c=0;c<max_y;c++) {
            board[c] = new int[max_x];
        }

        rowOne  = new int[max_x];
        rowZero = new int[max_x];
        topRow  = new int[max_x];
        midRow  = new int[max_x];
        botRow  = new int[max_x];

        rowUsage = new int[max_y];
    }
}
```

Тест на языке Java: Life (infinite)

```
import java.util.*;

public class infilife {
    static long wantedTime = 1000;
    static double[] benchmark = new double[] {234,251,255,257,260,249,232,221,209,199}; //
compared to C results
    static boolean useBenchmark = false;

    static String java_bmark_list = "";
    static long rateFactor;
    static int rerunMainLoopCount = 5; // has to be at least 3
    static boolean show_table = false;
    static int[] use_Sizes = new int[] {1,2,3,4,5,10,15,20,25,30};

    static int MAX_X = java.lang.Integer.MAX_VALUE;
    static int MAX_Y = java.lang.Integer.MAX_VALUE;
    static int MIN_X = java.lang.Integer.MIN_VALUE;
    static int MIN_Y = java.lang.Integer.MIN_VALUE;

    static final int PerC = 2;
    static final int LiveVal = 16;
    static final int cell_live_set = (1<<3) | 1<<(3+LiveVal) | 1<<(2+LiveVal);
    int[] board;
    int board_size;

    public static void main(String[] args) {
        if (args.length>0) wantedTime *= 10;
        double[] it_rates = new double[rerunMainLoopCount];
```

```
    if (useBenchmark) {
        System.out.println("Results are relative to baseline");
    } else {
        System.out.println("Results are in raw format");
    }
    System.out.println("Size\tbest\taverage\tworst");
    for (int c=0; c<use_Sizes.length; c++) {
        infilife life_ob = new infilife(use_Sizes[c]);
        double rate = life_ob.calcRate(1,wantedTime/100);
        long cycles = (long)(rate*wantedTime/10d);
        rate = life_ob.calcRate(cycles,wantedTime/10);
        cycles = (long)(rate*wantedTime);
        for (int d=0; d<5; d++) {
            rate = life_ob.calcRate(cycles,(long)(wantedTime*0.9));
            long check = (long)(rate*wantedTime);
            if (check > (cycles * 2)) {
                System.out.println("Adjusting "+cycles+" to "+check);
                cycles = check;
            }

            rate *= rateFactor;
            if (useBenchmark) {
                it_rates[d] = 100d * rate / benchmark[c];
            } else {
                it_rates[d] = rate;
            }
        }
        life_ob.calcPerformance(it_rates,(c==0));
    }
    System.out.println("Averages: "+java_bmark_list);
}

double calcRate(long cycles_per_time, long max_time) {
    long big_counts = 0;
    long time_taken = 0;
    long now = getTimeAtTick();
    long was = now;
    while (time_taken <= max_time) {
        big_counts ++;
        for (long i=0; i<cycles_per_time; i++) {
            calc_new();
        }
        now = System.currentTimeMillis();
        time_taken = now-was;
    }
    double rate = (double)(big_counts * cycles_per_time)/(double)time_taken;
    rateFactor = board.length;
    return rate;
}

static long getTimeAtTick() {
    long now = System.currentTimeMillis();
    final long was = now;
    while (was >= now) {
        now = System.currentTimeMillis();
    }
    return now;
}

private void calcPerformance(double[] it_rates, boolean is_first) {
     sortArray(it_rates);
    for (int rep=0; rep<(rerunMainLoopCount-3); rep++) {
        it_rates[1] += it_rates[rep+2];
    }
    it_rates[1] /= (rerunMainLoopCount-2);
    it_rates[2] = it_rates[rerunMainLoopCount-1];

    String[] outputs = new String[3];
    for (int rep=0; rep<3; rep++) {
        outputs[rep] = pretty_number(it_rates[rep]);
    }
    String output = "";
    for (int j=2; j>=0; j--) {
      if (j != 2) output += "\t";
      output += outputs[j];
    }
```

```
    if (!is_first) java_bmark_list += ",";
    java_bmark_list += outputs[1];
    System.out.println((board_size*5)+"\t"+output);
}

static private void sortArray(double[] array) {
    if (array.length < 2) return;
    boolean changed = true;
    while (changed) {
        changed = false;
        for (int c=1; c<array.length; c++) {
            if (array[c-1] > array[c]) {
                changed = true;
                double tmp = array[c];
                array[c] = array[c-1];
                array[c-1] = tmp;
            }
        }
    }
}

static private String pretty_number(double num) {
    double cutoff = 120;
    if (num > cutoff) return ""+(int)num;
    if (num < 0.001) return "0";
    double fac = 1d;
    while (num < cutoff) {
        num *= 10d;
        fac *= 10d;
    }
    int tt = (int)num;
    num = tt / fac;
    return ""+num;
}

public infilife(int size) {
    int alive_per_group = 5; // per glider
    int end = size*size * alive_per_group * PerC;
    int x_gap = 5;
    int y_gap = 5;
    board = new int[end];
    board_size = size;

    int set = 0;
    for (int col=0; col<size; col++) {
        int y = col*y_gap;
        for (int row=0; row<size; row++) {
            int x = row*x_gap;
            board[set+0] = x+0; board[set+1] = y+2;
            board[set+2] = x+1; board[set+3] = y+2;
            board[set+4] = x+2; board[set+5] = y+2;
            board[set+6] = x+2; board[set+7] = y+1;
            board[set+8] = x+1; board[set+9] = y+0;
            set += alive_per_group * PerC;
        }
    }
}

public void display(int min_x, int min_y, int max_x, int max_y) {
    char alive[][] = new char[max_y-min_y][max_x-min_x];
    for (int y=0; y<alive.length; y++) {
        char arow[] = alive[y];
        for (int x=0; x<arow.length; x++) {
            arow[x] = ' ';
        }
    }

    for (int i=0; i<board.length; i += PerC) {
        int x = board[i + 0];
        int y = board[i + 1];
        if (x <  min_x) continue;
        if (y <  min_y) continue;
        if (x >= max_x) continue;
        if (y >= max_y) continue;
        alive[y-min_y][x-min_x] = '#';
    }
```

```
    System.out.print("\t+");
    for (int x=0; x<alive[0].length; x++) System.out.print("-");
    System.out.println("+");
    for (int y=alive.length-1; y>=0; y--) {
        System.out.println((y+min_y)+"\t|"+(new String(alive[y])) + "|");
    }
    System.out.print("\t+");
    for (int x=0; x<alive[0].length; x++) System.out.print("-");
    System.out.println("+\n");
}

private void calc_new() {
    int alive = board.length / 2;
    int score[][] = new int[alive][];
    int alive_next = 0;
    for (int i=0; i<board.length; i += PerC) {
        int xpos_c = board[i + 0];
        int ypos_c = board[i + 1];

        int xpos_l = (xpos_c < MAX_X)? xpos_c-1: MIN_X;
        int xpos_r = (xpos_c > MIN_X)? xpos_c+1: MAX_X;
        int ypos_b = (ypos_c < MAX_Y)? ypos_c-1: MIN_Y;
        int ypos_t = (ypos_c > MIN_Y)? ypos_c+1: MAX_Y;

        alive_next += addVote(score,xpos_l,ypos_t, 1);
        alive_next += addVote(score,xpos_c,ypos_t, 1);
        alive_next += addVote(score,xpos_r,ypos_t, 1);
        alive_next += addVote(score,xpos_l,ypos_c, 1);
        alive_next += addVote(score,xpos_c,ypos_c, LiveVal);
        alive_next += addVote(score,xpos_r,ypos_c, 1);
        alive_next += addVote(score,xpos_l,ypos_b, 1);
        alive_next += addVote(score,xpos_c,ypos_b, 1);
        alive_next += addVote(score,xpos_r,ypos_b, 1);
    }
    //      System.out.println("AL:"+alive_next);

    int poi = 0;
    int new_board[] = new int[alive_next * PerC];
    for (int seti=0; seti<score.length; seti++) {
        int set[] = score[seti];
        if (set == null) continue;
        for (int i=0; i<set.length; i+=3) {
            int x = set[i+0];
            int y = set[i+1];
            int s = set[i+2];
            if (((1 << s) & cell_live_set) <= 0) continue;
            new_board[poi + 0] = x;
            new_board[poi + 1] = y;
            poi += PerC;
        }
    }

    board = new_board;
}

static private int addVote(int score[][],int x, int y, int val) {
    int hash = (x+y) ^ (x<<2 - y>>1);
    if (hash<0) hash = -hash;
    hash = hash % score.length;
    //      System.out.println(hash+" / "+score.length);
    int set[] = score[hash];
    if (set == null) {
        set = new int[3];
        set[0] = x;
        set[1] = y;
        set[2] = val;
        score[hash] = set;
        return 0;
    }

    for (int i=0; i<set.length; i+=3) {
        if ((set[i+0] != x) || (set[i+1] != y)) continue;
        int votes = set[i+2];
        int was_alive = 0;
        if (((1 << votes) & cell_live_set) > 0) was_alive = 1;
        votes += val;
        int now_alive = 0;
```

```
            if (((1 << votes) & cell_live_set) > 0) now_alive = 1;
            set[i+2] = votes;
            return (now_alive - was_alive);
        }

        int oldl = set.length;
        int nset[] = new int[oldl + 3];
        for (int i=0; i<oldl; i++) {
          nset[i] = set[i];
        }
        nset[oldl + 0] = x;
        nset[oldl + 1] = y;
        nset[oldl + 2] = val;
        score[hash] = nset;
        return 0;
    }
}
```

Тест на языке Java: Fibonacci

```
import java.util.*;

public class fib {
    static long wantedTime = 1000;
    static double[] benchmark = new double[]
{16837,15561,14976,14843,14807,14803,14805,14801};
    static boolean useBenchmark = false;

    static String java_bmark_list = "";
    static long rateFactor;
    static int[] fibNums = new int[] {5,7,10,12,15,20,22,25};
    static int rerunMainLoopCount = 5; // has to be at least 3

    public static void main(String[] args) {
        if (args.length>0) wantedTime *= 10;
        double[] it_rates = new double[rerunMainLoopCount];

        if (useBenchmark) {
            System.out.println("Results are relative to baseline");
        } else {
            System.out.println("Results are in raw format");
        }
        System.out.println("FibNum\tbest\taverage\tworst");

        for (int c=0; c<fibNums.length; c++) {
            int fib_num = fibNums[c];
            double rate = calcRate(fib_num,1,wantedTime/100);
            long cycles = (long)(rate*wantedTime/10d);
            rate = calcRate(fib_num,cycles,wantedTime/10);
            cycles = (long)(rate*wantedTime);
            for (int d=0; d<5; d++) {
                rate = rateFactor * calcRate(fib_num,cycles,0);
                if (useBenchmark) {
                    it_rates[d] = 100d * rate / benchmark[c];
                } else {
                    it_rates[d] = rate;
                }
            }
            calcPerformance(fib_num,it_rates,(c==0));
        }
        if (useBenchmark) {
            System.out.println("Averages: "+java_bmark_list);
        }
    }

    static int doFib(int fib) {
        if (fib < 2) return 1;
        return doFib(fib-1) + doFib(fib-2);
    }

    static double calcRate(int fib_num, long cycles_per_time, long max_time) {
        long big_counts = 0;
        long time_taken = 0;
        long now = getTimeAtTick();
        long was = now;
        while (time_taken <= max_time) {
            big_counts ++;
```

```
            for (long i=0; i<cycles_per_time; i++) {
                rateFactor = doFib(fib_num);
            }
            now = System.currentTimeMillis();
            time_taken = now-was;
        }
        double rate = (double)(big_counts * cycles_per_time)/(double)time_taken;
        return rate;
    }

    static long getTimeAtTick() {
        long now = System.currentTimeMillis();
        final long was = now;
        while (was >= now) {
            now = System.currentTimeMillis();
        }
        return now;
    }

    static private void calcPerformance(int fib_num, double[] it_rates, boolean is_first) {
        sortArray(it_rates);
        for (int rep=0; rep<(rerunMainLoopCount-3); rep++) {
            it_rates[1] += it_rates[rep+2];
        }
        it_rates[1] /= (rerunMainLoopCount-2);
        it_rates[2] = it_rates[rerunMainLoopCount-1];

        String[] outputs = new String[3];
        for (int rep=0; rep<3; rep++) {
            outputs[rep] = pretty_number(it_rates[rep]);
        }
        String output = "";
        for (int j=2; j>=0; j--) {
            if (j != 2) output += "\t";
            output += outputs[j];
        }
        if (!is_first) java_bmark_list += ",";
        java_bmark_list += outputs[1];
        System.out.println(fib_num+"\t"+output);
    }

    static private void sortArray(double[] array) {
        if (array.length < 2) return;
        boolean changed = true;
        while (changed) {
            changed = false;
            for (int c=1; c<array.length; c++) {
                if (array[c-1] > array[c]) {
                    changed = true;
                    double tmp = array[c];
                    array[c] = array[c-1];
                    array[c-1] = tmp;
                }
            }
        }
    }

    static private String pretty_number(double num) {
        double cutoff = 120;
        if (num > cutoff) return ""+(int)num;
        if (num < 0.001) return "0";
        double fac = 1d;
        while (num < cutoff) {
            num *= 10d;
            fac *= 10d;
        }
        int tt = (int)num;
        num = tt / fac;
        return ""+num;
    }
}
```

Тест на языке Java: FFT

```
import java.util.Random;

public class fft {
```

```
    static long wantedTime = 1000;
    static double[] benchmarks = new double[]
{775,991,1158,1260,1332,1219,957,273,232,183,134}; // compared to C results
    static boolean useBenchmark = false;

    static String java_bmark_list = "";
    static long rateFactor;
    static int rerunMainLoopCount = 5; // has to be at least 3
    static int[] use_Sizes = new int[] {3,4,5,6,8,10,12,13,14,15,16};

    double testArray[];
    public static double JDKtotal = 0.0;
    public static double JDKtotali = 0.0;

  /** Simple Test routine. */
  public static void main(String args[]){
      if (args.length>0) wantedTime *= 10;
      double[] it_rates = new double[rerunMainLoopCount];

      if (useBenchmark) {
          System.out.println("Results are relative to baseline");
      } else {
          System.out.println("Results are in raw format");
      }
      System.out.println("Size\tbest\taverage\tworst");

      for (int c=0; c<use_Sizes.length; c++) {
          fft fft_ob = new fft(1 << use_Sizes[c]);

          double rate = fft_ob.calcRate(1,wantedTime/100);
          long cycles = (long)(rate*wantedTime/10d);
          rate = fft_ob.calcRate(cycles,wantedTime/10);
          cycles = (long)(rate*wantedTime);
          for (int d=0; d<5; d++) {
              rate = fft_ob.calcRate(cycles,(long)(wantedTime*0.9));
              long check = (long)(rate*wantedTime);
              if (check > (cycles * 2)) {
                  System.out.println("Adjusting "+cycles+" to "+check);
                  cycles = check;
              }

              rate *= rateFactor;
              if (useBenchmark) {
                  it_rates[d] = 100d * rate / benchmarks[c];
              } else {
                  it_rates[d] = rate;
              }
          }
          System.out.print(use_Sizes[c]+"\t");
          fft_ob.calcPerformance(it_rates,(c==0));
      }
      System.out.println("Averages: "+java_bmark_list);
  }

    double calcRate(long cycles_per_time, long max_time) {
        long big_counts = 0;
        long time_taken = 0;
        double diff = 0d;
        long now = getTimeAtTick();
        long was = now;
        while (time_taken <= max_time) {
            big_counts ++;
            for (long i=0; i<cycles_per_time; i++) {
                diff += test(testArray);
            }
            now = System.currentTimeMillis();
            time_taken = now-was;
        }
        double rate = (double)(big_counts * cycles_per_time)/(double)time_taken;
        rateFactor = testArray.length;
        if (diff > 10) {
            System.out.println("Diff too big? "+diff);
        }
        return rate;
    }

    static long getTimeAtTick() {
```

```
        long now = System.currentTimeMillis();
        final long was = now;
        while (was >= now) {
            now = System.currentTimeMillis();
        }
        return now;
    }

       private void calcPerformance(double[] it_rates, boolean is_first) {
        sortArray(it_rates);
        for (int rep=0; rep<(rerunMainLoopCount-3); rep++) {
            it_rates[1] += it_rates[rep+2];
        }
        it_rates[1] /= (rerunMainLoopCount-2);
        it_rates[2] = it_rates[rerunMainLoopCount-1];

        String[] outputs = new String[3];
        for (int rep=0; rep<3; rep++) {
            outputs[rep] = pretty_number(it_rates[rep]);
        }
        String output = "";
        for (int j=2; j>=0; j--) {
            if (j != 2) output += "\t";
            output += outputs[j];
        }
        if (!is_first) java_bmark_list += ",";
        java_bmark_list += outputs[1];
        System.out.println(output);
    }

    static private String pretty_number(double num) {
        double cutoff = 120;
        if (num > cutoff) return ""+(int)num;
        if (num < 0.001) return "0";
        double fac = 1d;
        while (num < cutoff) {
            num *= 10d;
            fac *= 10d;
        }
        int tt = (int)num;
        num = tt / fac;
        return ""+num;
    }

    static private void sortArray(double[] array) {
        if (array.length < 2) return;
        boolean changed = true;
        while (changed) {
            changed = false;
            for (int c=1; c<array.length; c++) {
                if (array[c-1] > array[c]) {
                    changed = true;
                    double tmp = array[c];
                    array[c] = array[c-1];
                    array[c-1] = tmp;
                }
            }
        }
    }


/** Compute Fast Fourier Transform of (complex) data, in place.*/
public static void transform (double data[]) {
    transform_internal(data, -1);

    int JDKrange = data.length;
    for (int i=0; i<JDKrange; i++) {
     JDKtotal += data[i];
    }

}

/** Compute Inverse Fast Fourier Transform of (complex) data, in place.*/
public static void inverse (double data[]) {
    transform_internal(data, +1);
    // Normalize
    int nd=data.length;
```

```
    int n =nd/2;
    double norm=1/((double) n);
    for (int i=0; i<nd; i++)
      data[i] *= norm;

    for (int i=0; i<nd; i++) {
      JDKtotali += data[i];
    }

  }

    public fft(int size) {
        testArray = makeRandom(size);
    }

  /** Accuracy check on fft of data. Make a copy of data, Compute the fft, then
    * the inverse and compare to the original.  Returns the rms difference.*/
  public static double test(double data[]){
    int nd = data.length;
    // Make duplicate for comparison
    double copy[] = new double[nd];
    System.arraycopy(data,0,copy,0,nd);
    // Transform & invert
    transform(data);
    inverse(data);
    // Compute RMS difference.
    double diff = 0.0;
    for(int i=0; i<nd; i++) {
      double d = data[i]-copy[i];
      diff += d*d;
    }
    return Math.sqrt(diff/nd);
  }

  /** Make a random array of n (complex) elements. */
  public static double[] makeRandom(int n){
    int nd = 2*n;
    Random rnd = new Random(141432);
    double data[] = new double[nd];
    for(int i=0; i<nd; i++) {
      // Okay, this isn't exactly random, but it's simple and cross-platform, though the
actual numbers don't make a lot of difference.
      data[i]= (double)((i*i+1)%10) / 10.0;
    }
    return data;
  }

  /* ____________________________________________________________________ */

  protected static int log2 (int n){
    int log = 0;
    for(int k=1; k < n; k *= 2, log++);
    if (n != (1 << log))
      throw new Error("fft: Data length is not a power of 2!: "+n);
    return log;
  }

  protected static void transform_internal (double data[], int direction) {
    int n = data.length/2;
    if (n == 1) return;              // Identity operation!
    int logn = log2(n);

    /* bit reverse the input data for decimation in time algorithm */
    bitreverse(data) ;

    /* apply fft recursion */
    for (int bit = 0, dual = 1; bit < logn; bit++, dual *= 2) {
      double w_real = 1.0;
      double w_imag = 0.0;

      double theta = 2.0 * direction * Math.PI / (2.0 * (double) dual);
      double s = Math.sin(theta);
      double t = Math.sin(theta / 2.0);
      double s2 = 2.0 * t * t;

      /* a = 0 */
      for (int b = 0; b < n; b += 2 * dual) {
```

```
        int i = 2*b ;
        int j = 2*(b + dual);

        double wd_real = data[j] ;
        double wd_imag = data[j+1] ;

        data[j]   = data[i]   - wd_real;
        data[j+1] = data[i+1] - wd_imag;
        data[i]  += wd_real;
        data[i+1]+= wd_imag;
      }

      /* a = 1 .. (dual-1) */
      for (int a = 1; a < dual; a++) {
        /* trignometric recurrence for w-> exp(i theta) w */
        {
          double tmp_real = w_real - s * w_imag - s2 * w_real;
          double tmp_imag = w_imag + s * w_real - s2 * w_imag;
          w_real = tmp_real;
          w_imag = tmp_imag;
        }
        for (int b = 0; b < n; b += 2 * dual) {
          int i = 2*(b + a);
          int j = 2*(b + a + dual);

          double z1_real = data[j];
          double z1_imag = data[j+1];

          double wd_real = w_real * z1_real - w_imag * z1_imag;
          double wd_imag = w_real * z1_imag + w_imag * z1_real;

          data[j]   = data[i]   - wd_real;
          data[j+1] = data[i+1] - wd_imag;
          data[i]  += wd_real;
          data[i+1]+= wd_imag;
        }
      }
    }
  }


  protected static void bitreverse(double data[]) {
    /* This is the Goldrader bit-reversal algorithm */
    int n=data.length/2;
    for (int i = 0, j=0; i < n - 1; i++) {
      int ii = 2*i;
      int jj = 2*j;
      int k = n / 2 ;
      if (i < j) {
        double tmp_real    = data[ii];
        double tmp_imag    = data[ii+1];
        data[ii]   = data[jj];
        data[ii+1] = data[jj+1];
        data[jj]   = tmp_real;
        data[jj+1] = tmp_imag; }

      while (k <= j) {
        j = j - k ;
        k = k / 2 ; }
      j += k ;
    }
  }
}
```