На правах рукописи

АНТИПОВА Екатерина Владимировна

АВТОМАТИЗАЦИЯ ПРОЕКТИРОВАНИЯ АППАРАТНО-ЗАВИСИМЫХ ПРОГРАММНЫХ РЕАЛИЗАЦИЙ АВТОМАТНЫХ ДИАГРАММ

Специальность:
05.13.12. «Системы автоматизации проектирования (промышленность)»

АВТОРЕФЕРАТ
диссертации на соискание ученой степени
кандидата технических наук

Ульяновск — 2012

Работа выполнена на кафедре «**Вычислительная техника**» Ульяновского государственного технического университета.

Научный руководитель: доктор технических наук, профессор,
Негода Виктор Николаевич

Официальные оппоненты: **Егоров Юрий Петрович,**
доктор технических наук, профессор,
ФНПЦ ОАО «НПО "Марс"»,
главный специалист
Похилько Александр Федорович,
кандидат технических наук, профессор,
УлГТУ, кафедра «Системы автоматизации проектирования», профессор

Ведущая организация: ОАО «Ульяновское конструкторское бюро приборостроения»

Защита состоится «26» декабря 2012 года в 12 часов 00 минут на заседании диссертационного совета Д212.277.01 при Ульяновском государственном техническом университете, ауд. 211.

С диссертацией можно ознакомиться в научной библиотеке Ульяновского государственного технического университета.

Автореферат разослан 24 ноября 2012

Ученый секретарь
Диссертационного совета
д.т.н., профессор В. И. Смирнов

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность проблемы. При создании программного обеспечения встроенных систем широко применяются диаграммы состояний как средство определения алгоритмов функционирования. В последние 15 лет активно развиваются средства проектирования, которые позволяют автоматически генерировать программы из диаграмм состояний на основе технологии автоматного программирования. Этим занимаются Зюбин В.Е, Кузнецов Б.П, Любченко В.С., Страбыкин А.Д., Чунихин О.Ю., Шалыто А.А. и его ученики, Дж. Э. Хопкрофт, Р. Мотвани, Дж. Д. Ульман, А.В. Ахо и др.. Результатом этих исследований являются методы, алгоритмы и программные средства реализации автоматных диаграмм, обеспечивающие автоматизацию проектирования логической части программы управления.

В то же время в процесс автоматизированного проектирования встроенных систем вовлекается аппаратная часть систем управления. К сожалению, учет проектных решений именно в этой части в существующих программных средствах поддержки автоматного программирования не поддерживается. Варьирование проектных решений в аппаратной части требует модификации самих автоматных диаграмм, а используемые при этом механизмы трансформации не автоматизированы. Поддержка трассировки диаграмм в существующих системах не обеспечивает активное прототипирование, охватывающее значительную часть этапов проектирования, для которых вначале работают исключительно имитаторы процессов ввода-вывода, а в дальнейшем активно вовлекается соответствующая аппаратура. Поэтому исследования, направленные на автоматизацию аппаратно-зависимых программных реализаций автоматных диаграмм являются актуальными.

Область исследования – автоматизация проектирования систем логического управления.

Объект исследования – автоматизация проектирования автоматных диаграмм.

Предмет исследования – автоматизация проектирования аппаратно-зависимых программных реализаций диаграмм состояний.

Целью диссертационной работы является разработка моделей и средств автоматизации проектирования аппаратно-зависимых программных реализаций автоматных диаграмм, предназначенных для ускорения процесса разработки программного обеспечения встроенных систем за счет вовлечения аппаратно-зависимых проектных решений на ранних этапах разработки с целью получения наиболее эффективных реализаций.

Задачи диссертационной работы. Для достижения цели диссертационного исследования необходимо решить следующие задачи:

1. Разработать подход к процессу проектирования и базовую модель процесса проектирования аппаратно-программных реализаций автоматных диаграмм, предназначенную для описания и реализации статических, динамических и аппаратно-зависимых аспектов процессов управления.

2. Разработать и реализовать механизмы структурных и аппаратно-зависимых трансформаций автоматных диаграмм.

3. Разработать и реализовать алгоритмы автоматической генерации программ, учитывающие статические, динамические и аппаратно-зависимые аспекты процесса управления.

4. Разработать программную систему поддержки автоматизированного проектирования аппаратно-зависимых программных реализаций автоматных диаграмм.

Методы исследования. При решении поставленных задач использованы методы теории автоматов, теории графов, объектно-ориентированного программирования, модельно-ориентированного проектирования, теории компиляторов и языков программирования.

Научная новизна проведенных исследований заключается в следующем:

1. Разработанный подход к автоматизированному проектированию аппаратно-зависимых программных реализаций автоматных диаграмм позволяет получать программы на любом языке программирования для любой целевой платформы с учетом схемы подключения устройств (источников сигналов, средств отображения и объектов управления) и поддерживает активное прототипирование на различных этапах проектирования.

2. Дано определение автоматной модели как совокупности диаграммы классов, диаграмм состояний и диаграммы подключений, которые позволяют учесть все аспекты процесса управления: статические (для определения структуры программы), динамические (для разработки алгоритмов поведения программы) и аппаратно-зависимые, определяющие порядок взаимодействия программы с внешними устройствами и факторы реального времени.

3. Разработана модель процесса автоматизированного проектирования аппаратно-зависимых программных реализаций автоматных диаграмм, отличающаяся от известных моделей и методов поддержкой двух видов трансформаций: трансформаций самих исходных диаграмм на основе использования расширенной нотации языка UML с учетом проектных решений аппаратной части, а также трансформаций структурно-функциональной организации порождаемого исходного кода программ.

4. Разработана система языков спецификаций исходных данных и продуктов автоматической генерации аппаратно-зависимых программных реализаций автоматных диаграмм, а также система правил трансформации автоматных диаграмм, учитывающая изменения в проектных решениях аппаратной части создаваемой системы управления.

5. Разработана подсистема САПР, осуществляющая поддержку предложенного подхода, предоставляющая возможность выбора целевой платформы и языка программирования, механизмов трансформации автоматных диаграмм и автоматической генерации автоматных программ с целью получить наиболее эффективную реализацию, а также обеспечивающая активное прототипирование на различных этапах проектирования в режиме отладки спецификаций трансформированных диаграмм.

Практическая ценность полученных результатов состоит в том, что созданные средства САПР повышают производительность труда проектировщика систем логического управления (на 30-35% в рамках проектных задач экспериментальной части диссертации). Использование разработанных средств САПР в учебном процессе позволяет сократить время изучения основ теории автоматов и автоматного программирования в 2-3 раза.

Предложенные методы проектирования и реализации автоматных моделей позволяют повысить эффективность разработки программного обеспечения для встроенных систем в силу использования модельно-ориентированного подхода и формализации перехода от автоматной модели к программному коду. Возможность выбора механизма генерации программного кода дает преимущества подстройки программной реализации под конкретную задачу и параметры целевой среды. Возможность выбора целевой платформы дает преимущества быстрого переноса программы на другую целевую платформу. Возможность использования эквивалентных преобразований диаграмм состояний порождает ряд вариантов алгоритма для выбора наиболее эффективной реализации в каждом конкретном случае.

Результаты данной работы могут быть использованы при проектировании систем управления в различных целях.

Достоверность и эффективность научных положений, выводов и практических рекомендаций, полученных в диссертации, подтверждается доказательствами, корректным обоснованием постановок задач, точной формулировкой критериев, компьютерным моделированием и результатами практического использования средств САПР, созданных на основе теоретических результатов работы.

Внедрение результатов. Результаты, полученные в диссертации, используются в компании *Тауруна* (г. Ульяновск) при разработке программного обеспечения для микроконтроллеров и в учебном процессе УлГТУ.

Основные положения, выносимые на защиту:

1. Подход к проектированию аппаратно-зависимых программных реализаций автоматных диаграмм, отличающийся от известных подходов возможностью получать программный код на любом языке программирования для произвольной целевой платформы с учетом аппаратных аспектов системы.

2. Определение автоматной модели как совокупности диаграммы классов, диаграмм состояний и диаграммы подключений, которые позволяют учесть все аспекты разрабатываемой программы управления встроенной системы: статические (для определения структуры программы управления), динамические (для разработки алгоритмов поведения программы управления) и аппаратно-зависимые, определяющие порядок взаимодействия программы микроконтроллера с внешними устройствами.

3. Модель процесса автоматизированного проектирования аппаратно-зависимых программных реализаций автоматных диаграммы, использующая расширенную нотацию языка UML для описания автоматных моделей и отличающаяся от известных моделей и методов рядом особенностей.

4. Система языков спецификаций исходных данных и продуктов генерации программно-аппаратных реализаций автоматных диаграмм, а также система эквивалентных преобразований автоматных диаграмм.

5. Подсистема САПР, предназначенная для поддержки предложенного подхода, предоставляющая возможность отладки автоматной модели на различных этапах проектирования, выбора целевой платформы и языка, механизмов трансформации и генерации автоматных программ.

Публикации. По теме диссертации опубликовано 10 печатных работ, в том числе 2 – в изданиях, рекомендованных ВАК РФ.

Структура и объем диссертации. Диссертационная работа состоит из введения, четырех глав, заключения, списка литературы и пяти приложений. Основное содержание изложено на 193 страницах, включая 63 рисунка и 28 таблиц. Список литературы включает 273 наименования.

СТРУКТУРА ДИССЕРТАЦИОННОЙ РАБОТЫ

Во введении обоснована актуальность темы диссертации, сформулирована цель, основные задачи исследований, определены объект и предмет исследований, основные научные результаты.

В первой главе приведен обзор и анализ программных средств автоматизации проектирования реализаций автоматных диаграмм, обзор и анализ специализированных программных средств разработки программ для промышленных микроконтроллеров. Рассмотрены различные языки описания автоматных диаграмм, методы автоматного программирования, механизмы генерации автоматных программ в различных средах программирования.

Автоматная модель позволяет полностью описать алгоритм функционирования встроенной системы. Диаграммы состояний UML основаны на расширении модели конечного автомата *Statecharts,* введенном Д. Харелом в 1987, и являются одним из наиболее гибких и мощных средств для описания автомата, предоставляющим ряд возможностей: описание автоматной диаграммы, средства декомпозиции (композитные состояния), расширенная спецификация переходов и состояний. При этом возможно применение стандартных алгоритмов преобразований теории автоматов.

Широко распространен подход к программированию систем логического управления, основанный на стандарте программирования контроллеров и схемотехнических устройств МЭК-61131-3, в котором описываются языки программирования, созданные на основе наиболее популярных языков программирования для контроллеров. Всего их 5: SFC (Sequential function chart), IL (Instruction list), ST (Structured text), LD (Ladder diagram), FBD (Function block diagram). Однако описать конечный автомат средствами языков МЭК довольно сложно в силу особенностей их формализма.

Композитные состояния диаграмм состояний представляют собой мощный инструмент декомпозиции автоматов. Но гибкость использования композитных состояний при проектировании управляющего устройства порождает сложность реализации некоторых моментов. В частности, неясно, как должен отображаться

на диаграмме переход между состояниями различного уровня вложенности при свернутых композитных состояниях (сквозной переход), а также в каком порядке выполняются действия при подобных переходах. В формализме автоматного программирования по технологии SWITCH (разрабатывается с 1991 года в СПбГУ ИТМО) в проекте Unimod композитные состояния заменены вложенными автоматами. И это исключает возможность переходов между состояниями различного уровня вложенности. Хотя в более ранних работах есть упоминание о преобразовании исходного графа и переводе вложенных состояний в составе композитного в граф более высокого уровня.

В работах по проекту ЗЕБРА (Чунихин О.Ю., г. Новосибирск) приведена формальная модель диаграмм состояний UML, в том числе формально определен композитный переход. Определена последовательность активаций состояний при переходе в композитное состояние и во вложенное состояние. Но композитный переход предназначен для работы с параллельными регионами, а в дальнейших работах автора упоминается, что от их поддержки пришлось отказаться по причине сложности и неэффективности реализации.

Приведенные формальные модели не описывают, как обрабатывать сквозные переходы во внутренние состояния композитных состояний.

Далее в данной главе рассмотрены механизмы генерации программного кода из диаграмм состояний: отображение множества актуальных состояний в аргумент оператора выбора, отображение множества актуальных состояний во множество меток перехода, отображение множества переходов в общий список спецификаций автомата, отображение множества актуальных состояний в объекты класса State, отображение множества переходов в сегментированное множество переходов автомата.

Среди программных сред проектирования автоматных диаграмм, поддерживающие автоматическую генерацию программ, отмечены следующие: Astade (инструмент UML моделирования, генерирующий программы на C++), конвертор Visio2Switch (предназначен для автоматической генерации кода на языке C по автоматным графам, разработанным в MS Visio в соответствии с требованиями SWITCH-технологии), инструментальное средство UniMod (предназначено для создания «исполняемых» графов переходов для платформы Eclipse), Finite State Machine Editor (включает в себя компоненту для создания и редактирования графов переходов автоматов, конвертер графов переходов, сохраненных в XML-формате в языки C++ и Python, компоненту для визуальной отладки графов переходов), инструментальное средство MetaAuto (предназначено для разработки проектов в автоматной парадигме), State Machine Compiler (инструмент, который сопоставляет конечные автоматы и объекты, генерирует программы из диаграмм состояний по образцу проектирования State на языках C++, Java, tcl, VB, C#), StarUML (предназначен для быстрой, гибкой, расширяемой разработки), Visual Paradigm for UML (полнофункциональное средство разработки приложений, поддерживающее все виды диаграмм UML 2.0, SysML и ER-диаграммы), MatLab Simulink (мощный инструмент, позволяющий разрабатывать диаграммы состояний и генерировать из них программный код на C для многих

микроконтроллеров). Каждая из перечисленных систем обладает рядом достоинств, однако ни одна из них не позволяет проектировать программу для произвольного микроконтроллера и решать низкоуровневые проблемы на ранних этапах проектирования.

Существуют узкоспециализированные программные инструменты для программирования микроконтроллеров на языках МЭК-61131-3, поддерживающие аппаратно-зависимое проектирование для микроконтроллеров одного-двух производителей. Только несколько из них (например, Siemens Simatic Step7 и Actum Realizer) поддерживают разработку алгоритма функционирования программы микроконтроллера в автоматных диаграммах с учетом особенностей целевой архитектуры. Однако, реализация аппарата автоматных диаграмм в них весьма скудна (в Actum Realizer нет вложенных состояний), автоматизация рутинных операций остается на низком уровне (в Simatic Step 7 приходится вручную вводить данные в таблицу символов), а также ориентированность на очень ограниченный класс целевых платформ не позволяет в полной объеме решить поставленные задачи.

Во второй главе описаны подход и схема процесса автоматизированного проектирования аппаратно-зависимых программных реализаций автоматных диаграмм, языки спецификаций автоматных диаграмм в рамках предлагаемого подхода, варианты структурно-функциональной организации порождаемого программного кода, механизмы трансформаций автоматных диаграмм, спецификации шаблонной целевой среды, протоколы трассировки и взаимодействия автоматной модели и целевого устройства.

Автоматная модель представляет собой формализованное описание проектируемой системы, статические аспекты которой отображаются на диаграмме классов, динамические аспекты прорабатываются во множестве диаграмм состояний, а взаимодействие различных физических компонентов системы представлено на диаграмме подключений.

Автоматизация проектирования аппаратно-зависимых программных реализаций автоматных диаграмм ориентируется на поддержку проектного процесса, обобщенная схема которого представлена на рисунке 1.

На рисунке 1 пунктиром обведены компоненты, предложенные ранее в работах других авторов, но переработанные и расширенные в данном исследовании. Жирным контуром обозначены компоненты, отсутствующие в других системах. Буквами А и Я обозначены аппаратные и языковые аспекты, в большей мере влияющие на соответствующий этап проектирования.

Как видно из рисунка 1, процесс опирается на классический цикл автоматизированного проектирования.

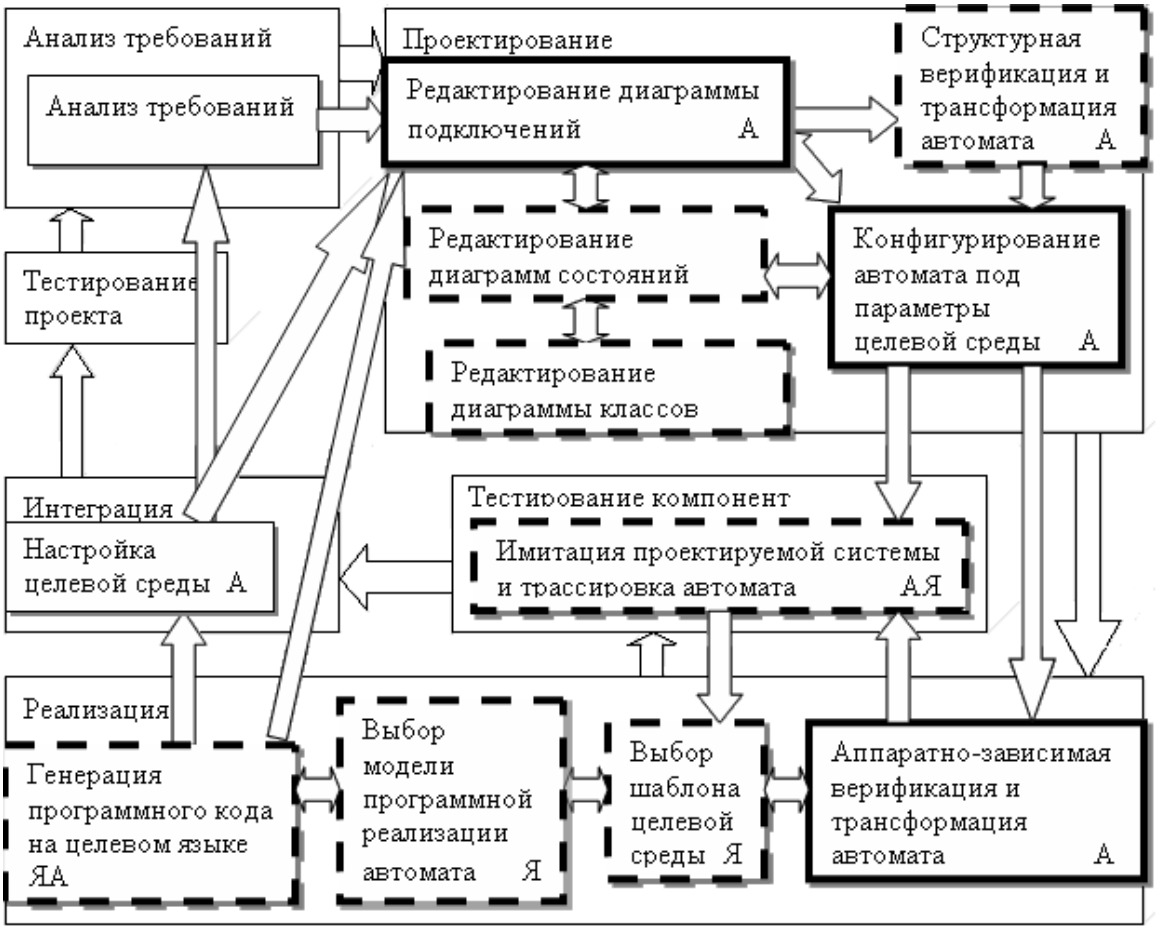

Рис. 1. Обобщенная схема проектного процесса автоматизированного проектирования программно-аппаратных реализаций автоматных диаграмм

Предлагаемый подход к проектированию программно-аппаратных реализаций автоматных диаграмм состоит из следующих положений:

1. Производится первичный анализ требований к реализуемой системе, на основе которого формируются множества состояний управляющего устройства и множества переходов между состояниями системы (без уточнения аппаратно-зависимых спецификаций).

2. Разрабатывается диаграмма подключений и производится первичное конфигурирование макета проектируемой системы (выбор и настройка микроконтроллера, объектов управления и т.д.)

3. В отличие от других средств разработки программного обеспечения встроенных систем, в результате разработки диаграмм подключений автоматически формируется множество аппаратно-зависимых компонентов для диаграммы состояний.

4. В процессе разработки диаграммы подключений определяются параметры программы управления – интервал дискретности, источник тактовой частоты с учетом аппаратных возможностей целевой платформы.

5. Для определения структуры программы управления разрабатывается диаграмма классов.

6. Для определения алгоритмов поведения программы управления разрабатывается диаграмма состояний.

7. После разработки диаграммы состояний производится автоматический анализ и, при необходимости, трансформация диаграммы состояний. Трансформация позволяет получить более компактные диаграммы состояний.

8. В отличие от известных подходов к разработке программного обеспечения встроенных систем, в предлагаемом подходе возможна аппаратно-зависимая трансформация автоматных диаграмм, что позволяет быстро переходить от одной аппаратной реализации к другой при необходимости.

9. Раннее прототипирование поддерживается процессами визуальной трассировки, которая отличается от известных реализаций трассировки диаграмм состояний интерпретацией кода встроенных функций и учетом аппаратных элементов в ходе отладки.

10. В предлагаемом подходе генерация программного кода происходит не в строго предопределенный язык программирования, а посредством шаблонов целевой среды, где в дальнейшем будет компилироваться код. Это позволяет генерировать программы на любом языке и для любой целевой среды, для которых возможно создать шаблон.

11. При генерации программного кода предоставляется выбор варианта структурно-функциональной организации порождаемой программы, что сужает возможности варьировать параметры создаваемого проектного решения.

Рассмотренный подход к автоматизации проектирования программно-аппаратных реализаций автоматных диаграмм описывает технологию создания программного обеспечения для микроконтроллеров.

Микроконтроллер и различные устройства, с которыми микроконтроллер взаимодействует в процессе работы программы, размещаются на диаграмме подключений. Общий вид диаграммы подключений представлен на рисунке 2.

Диаграмма подключений предназначена для определения количества сопряженных с микроконтроллером устройств, а также для автоматического распределения однотипных устройств по портам ввода-вывода. На этапе разработки диаграммы подключений определяется тип функционирования системы: событийный, дискретный или гибридный. Автоматически генерируются множества входных и выходных портовых переменных и определяются их значения, и множества входных сигналов.

Еще одной функцией диаграммы подключений является автоматическая минимизация подключений однотипных устройств и переопределение значений входных и выходных переменных, если контактов микроконтроллера не достаточно. При событийном принципе работы микроконтроллера это становится нетривиальной задачей.

Таким образом, диаграмма подключений решает следующие задачи:

1. Определение или переопределение целевой платформы (микроконтроллера) из расширяемой библиотеки.

2. Определение или переопределение направленности выводов микроконтроллера (входные, выходные и двунаправленные контакты).

3. Определение или переопределение типа функционирования проектируемой системы (событийный, дискретный или гибридный).

4. Выбор сопряженных устройств из расширяемой библиотеки.

5. Определение или переопределение подключения сопряженных устройств к контактам микроконтроллера.

6. Автоматическая генерация входных сигналов, входных и выходных портовых переменных, исходя из подключений устройств к микроконтроллеру.

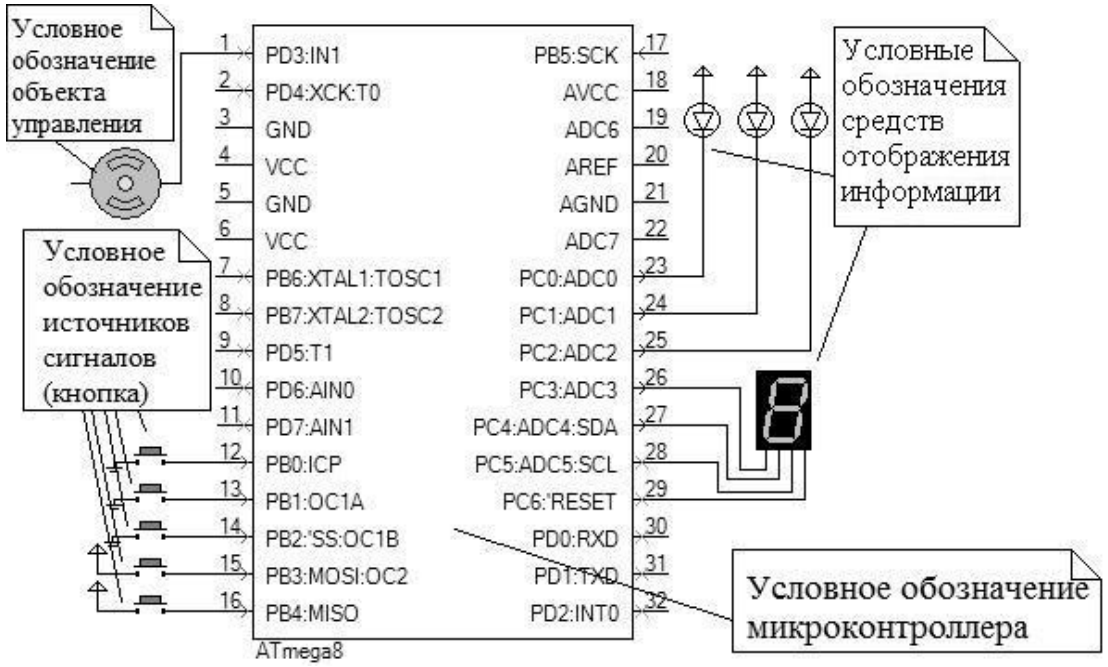

Рис. 2. Общий вид диаграммы подключений

Таким образом, подавляющее большинство низкоуровневых (аппаратно-зависимых) проблем разработки программ для микроконтроллеров решается на высоком уровне в ходе визуального проектирования.

Статические аспекты процесса управления описываются при помощи диаграммы классов UML. Общий вид диаграммы классов для описания структуры автомата приведен на рисунке 3.

Класс содержит, как минимум, один метод со стереотипом поведения «automaton», в котором реализуется поведение программы управления. Диаграмма классов создается в основном автоматически в процессе разработки диаграмм состояний и диаграммы подключений. Все входные сигналы, функции, входные и выходные портовые переменные генерируются при разработке диаграммы подключений, а продукты структурной трансформации специфицируются в процессе разработки диаграммы состояний. Вручную в диаграмму классов добавляются только функции и переменные, не обозначенные явно в диаграмме состояний.

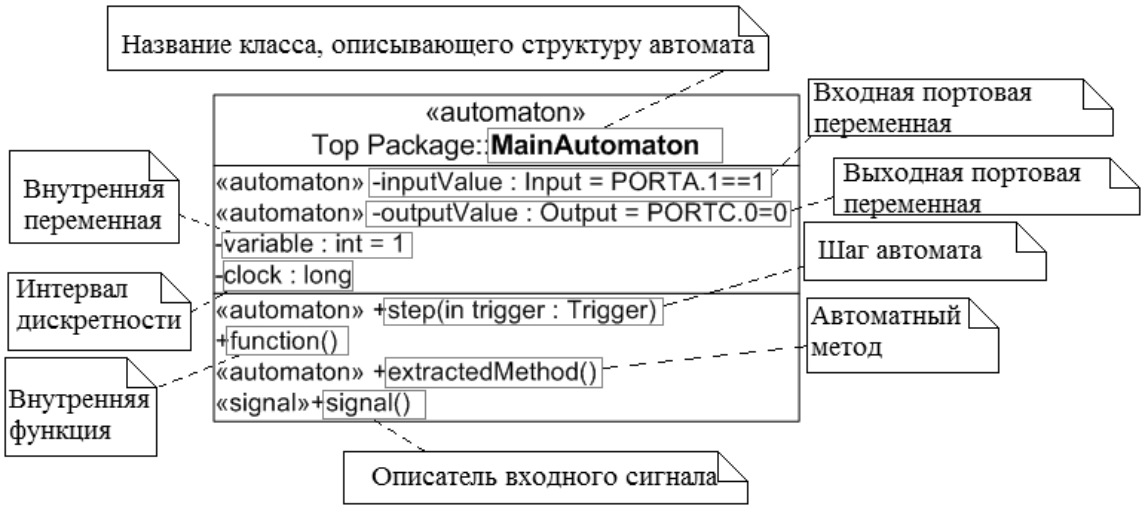

Рис. 3. Общий вид диаграммы классов для описания автомата

На рисунке 4 приведен общий вид диаграммы состояний для описания поведения программы управления.

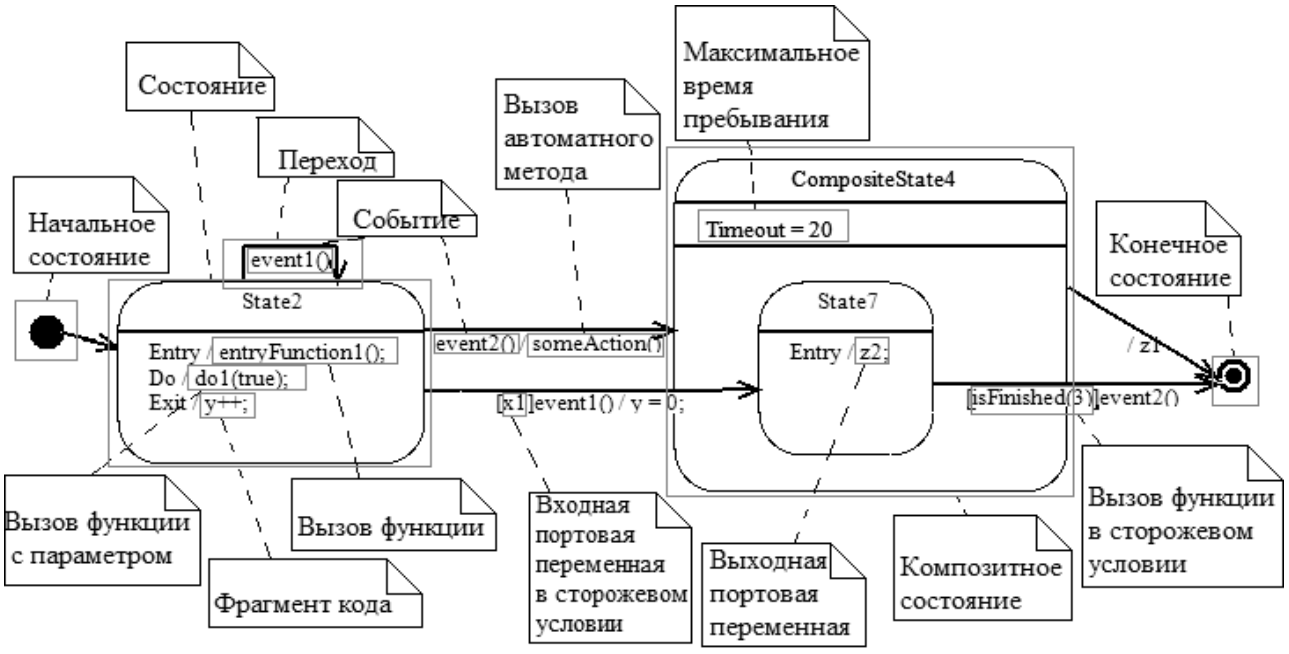

Рис. 4. Общий вид диаграммы состояний

Логико-алгебраическая модель спецификаций автоматных диаграмм представляется в виде множества $M = \{CD, SD, \{AD_i\}, F\}$, где CD – диаграмма подключений, SD – диаграмма классов, F – множество операций, а $\{AD_i\}$ – множество автоматных диаграмм. Состояние S_i из множества состояний S^n автоматной диаграммы AD_n обозначается S^n_i. Множество переходов автоматной диаграммы AD_n обозначается T^n.

Логико-алгебраическая модель диаграммы подключений представляет собой множество $CD = \{Chip, \{Device_i\}, Model\}$, где $Chip$ – формальное описание

микроконтроллера, $\{Device_i\}$ – множество сопряженных устройств, *Model* – множество настроек проекта.

Формальная модель микроконтроллера представляет собой множество $Chip = \{Name, Clock, IntSRAM, ExtSRAM, EEPROM, ProgFlash, \{Port_i, i=0..N\}, \{ChipPin_j, j=1..M\}, \{Interruption_k, k=0..K\}, \{Timer_t, t=0..T\}\}$, включающее в себя формальные описания множеств портов ввода-вывода, контактов, прерываний и встроенных таймеров. Модель сопряженного устройства описывается множеством $Device_i = \{Name, Description, View, DeviceType, DeviceCategory, \{DeviceProperty_j, j=0..N\}, \{Pin_j, j=1..M\}, \{State_k, k=1..K\}, \{Connection_t, t=1..T\}, \{variable_v, v=0..V\}, \{function_f, f=0..F\}, \{area_r, r=0..R\}, \{subDevice_s, s=0..S\}\}$, включающего в себя тип и категорию устройства, а также формальные описания множеств состояний, контактов, соединений, внутренних переменных и функций, интерактивных областей, вложенных устройств. Устройства относятся к одной из категорий (источник событий, устройство отображения, объект воздействия) и к одному из типов (светодиод, дисплей, кнопка, генератор, электромотор, регулятор и т.д.) Состояние устройства зависит от комбинации значений на входных контактах, на выходные контакты выводятся соответствующие значения. Значения на выходах устройства могут зависеть от значений внутренних переменных и выполнения внутренних функций.

Множество операций представляет собой множество $F = F_{\text{function}} \cup F_{\text{code}} \cup \cup F_{\text{input}} \cup F_{\text{output}} \cup F_{\text{signal}} \cup F_{\text{automaton}}$, где F_{function} – множество встроенных подпрограмм, описанных на промежуточном языке программирования, F_{code} – множество фрагментов кода на промежуточном языке программирования, F_{input} – множество входных портовых переменных, F_{output} – множество выходных портовых переменных, F_{signal} – множество входных сигналов, $F_{\text{automaton}}$ – множество программных модулей, поведение которых описывается отдельной автоматной диаграммой. В качестве промежуточного языка выбран C, как наиболее распространенный язык для программирования микроконтроллеров.

Входные портовые переменные используются для проверки состояния на отдельных входных портах или битах портов и специфицируются состоянием входных битов портов ввода-вывода микроконтроллера. Выходная портовая переменная используется для записи новых значений в выходные порты или определенные биты выходных портов. Входные сигналы соотносятся с множеством прерываний микроконтроллера.

Простое состояние описывается множеством S_s:simple = {Id, ParentId, Description, F_{entry}, F_{do}, F_{exit}, Timeout, T_{in}, T_{out}}, где Id – идентификатор состояния, ParentId – идентификатор композитного состояния, в которое вложено данное состояние, Description – описание, F_{entry} – множество операций, выполняемых при входе в состояние ($F_{\text{entry}} \subseteq F_{\text{function}} \cup F_{\text{code}} \cup F_{\text{output}} \cup F_{\text{automaton}}$), F_{do} – множество операций, вызываемых циклически до тех пор, пока не происходит выход из состояния ($F_{\text{do}} \subseteq F_{\text{function}} \cup F_{\text{code}} \cup F_{\text{output}} \cup F_{\text{automaton}}$), F_{exit} – множество операций, вызываемых непосредственно перед выходом из состояния ($F_{\text{exit}} \subseteq F_{\text{function}} \cup F_{\text{code}} \cup \cup F_{\text{output}} \cup F_{\text{automaton}}$), Timeout – постоянная величина или формула, определяющая

максимальное время пребывания в состоянии (Timeout $\in F_{\text{function}} \cup F_{\text{code}}$), T_{in}, T_{out} – множества входящих и исходящих переходов.

Композитное состояние описывается множеством S_c:composite = {Id, Type, ParentId, Description, F_{entry}, F_{do}, F_{exit}, Timeout, T_{in}, T_{out}, $S_{\text{substates}}$, S_{enryref}, S_{exitref}}. Отличие от описания простого состояния состоит во множестве вложенных состояний $S_{\text{substates}}$, множествах ссылок на точки входа S_{enryref} и ссылок на точки выхода S_{exitref}. Непосредственное вложение состояния S^m_j в композитное состояние S^l_k обозначается $S^m_j \in S^l_k$. Произвольный уровень вложенности обозначается $S^m_j \in^* S^l_k$. Точка входа и точка выхода композитного состояния S^p_q обозначается $S^l\text{entry}S^p_q$ и $S^l\text{exit}S^p_q$ соответственно. Соответствующие им ссылки на точки входа и точки выхода обозначаются $S^l_r\text{entry}S^p_q$ и $S^l_r\text{exit}S^p_q$.

Переход $T^n_i \in T^n$ описывается множеством {Id, ParentId, guard, event, F_{action}}, где Id – идентификатор перехода; ParentId – идентификатор композитного состояния, которому принадлежит переход; guard – сторожевое условие, принимающее в качестве аргументов элементы множеств F_{input}, F_{function}, $F_{\text{automaton}}$ или F_{code}; event $\in F_{\text{signal}}$; $F_{\text{action}} \subseteq F_{\text{function}} \cup F_{\text{code}} \cup F_{\text{output}} \cup F_{\text{automaton}}$.

Переход из состояния S^n_i в состояние S^m_j по условию d_x и с выполнением операций a_y обозначается $S^n_i\,(d_x/a_y) \to S^m_j$, где $d_x \in$ guard × event.

Сквозным переходом называется непосредственный переход между состояниями различного уровня вложенности и всегда состоит из сегментов. Разложение на сегменты обусловлено необходимостью детерминировать моменты выполнения действий при входе и выходе композитных состояний и действий при входе и выходе вложенных состояний, момент выполнения действий на переходе, а также работу с ограничениями по времени пребывания внутри композитного состояния.

Для сквозного перехода $S^n_i\,(d_x/a_y) \to S^m_j$, где $n \neq m$ или одно или оба из состояний S^n_i и S^m_j являются композитными, возможны следующие варианты разложения данного сквозного перехода на сегменты.

1. Сквозной переход во вложенное состояние более глубокого уровня вложенности. Если $\exists S^n_k : S^m_j \in^* S^n_k$, и S^l – диаграмма, непосредственно вложенная в S^n_k, то $S^n_i\,(d_x/a_y) \to S^m_j = S^n_i(d_x) \to S^l_r\text{entry}S^n_k + S^l\text{entry}S^n_k\,(d_x/a_y) \to S^m_j$.

2. Вход в композитное состояние. Если $\exists S^k_l : S^k_l \in S^m_j$, то $S^n_i(d_x/a_y) \to S^m_j = S^n_i\,(d_x/a_y) \to S^k_0$.

3. Выход из явно указанного вложенного состояния. Если $\exists S^m_k : S^n_i \in^* S^m_k$, и S^l – диаграмма, непосредственно вложенная в S^m_k то $S^n_i\,(d_x/a_y) \to S^m_j = S^n_i\,(d_x) \to \to S^l\text{exit}S^m_k + S^l_r\text{exit}S^m_k\,(d_x/a_y) \to S^m_j$.

4. Выход из композитного состояния. Если $\exists S^l_k : S^l_k \in S^n_i$, то $\forall t : t = 1..T$, $S^l_t \in S^n_i$, $S^n_i\,(d_x/a_y) \to S^m_j = \{S^l_t\,(d_x) \to S^l\text{exit}S^n_i,\ t = 1..T\} + S^l_r\text{exit}S^n_i\,(d_x/a_y) \to S^m_j$.

Для отображения множества автоматных диаграмм *AD* в программный код описаны алгоритмы преобразования элементов диаграмм в языковые конструкции языка программирования, т.е. шаблон целевой среды, который представляет собой множество параметризованных конструкций целевого языка программирования $Template_{\text{i}}$ = {Id, {P_{j}, j = 1..m}, Code}. Каждая такая конструкция имеет свой

идентификатор, список параметров и фрагмент программного кода на целевом языке программирования, использующий параметры для подстановки. Библиотека шаблонов является расширяемой.

Для реализации трассировки автоматной модели разработаны спецификации входных и выходных протоколов. Входной протокол трассировки автомата можно представить множеством *Trace* элементов вида $trace_i = \{n, t, f\}$, где n – номер такта, t – модельное время, $f \in F_{signal}$. Выходной протокол может быть кратким и расширенным. Краткий выходной протокол описывается множеством $\{n, t, f, s_{from}, s_{to}\}$, где n – номер такта, t – модельное время, $f \in F_{signal}$ – принятый сигнал, $s_{from}, s_{to} \in S$ – исходное и целевое состояния.

Расширенный выходной протокол трассировки, помимо выше описанных данных, содержит значения всех портов ввода-вывода управляющего устройства, а также значения внутренних переменных автомата.

Кроме того, разработаны и реализованы механизмы трансформации диаграмм состояний. Трансформация может быть структурной или аппаратно-зависимой. К структурным механизмам трансформации относятся инкапсуляция реализации, выделение в композитное состояние или автоматный метод возвратных частей автомата, объединение в композитное состояние групп состояний с общими внутренними спецификациями. Аппаратно-зависимая трансформация применима в случаях, когда разработанный комплекс автоматных диаграмм не может быть реализован в целевой платформе в текущей конфигурации. В рамках исследования разработаны и реализованы механизмы минимизации подключений и механизмы обработки взаимоисключающих и взаимозависимых сигналов.

На основе описанных выше множеств построена БНФ-нотация языков спецификаций исходных данных и продуктов генерации.

Для получения различных вариантов структурно-функциональной организации порождаемого кода реализованы алгоритмы генерации на основе технологии switch, с применением образца проектирования Statemachine, два табличных метода с различной организацией описания автомата в таблице (общий список переходов и структурированный по множеству условий переходов список). Реализован также способ структурно-функциональной организации автоматных программ с использованием меток перехода.

В третьей главе описывается архитектура программной системы поддержки автоматизированного проектирования программно-аппаратных реализаций автоматных диаграмм согласно предложенному подходу.

Программная среда содержит 10 компонентов: модель программно-аппаратных реализаций автоматных диаграмм, графический редактор диаграмм, свойства графических элементов диаграмм, редактор и интерпретатор встроенных функций, определение целевой платформы, определение целевой среды и генерация программного кода, трансформация автоматной модели, имитационное тестирование, отображение элементов автоматной модели, трассировка автоматной модели. Разработано 10 диаграмм классов для каждого компонента

системы, всего 136 классов и более 300 ассоциаций. Реализация программной среды составляет более 30000 строк кода на языке C#.

Программная среда EBA предоставляет средства для разработки диаграмм подключений, классов и состояний, импорта диаграмм состояний и классов, разработанных в других системах, выполнения структурной и аппаратно-зависимой трансформации автоматных моделей, содержит инструменты визуальной трассировки и имитации автоматных моделей, а также генерации программных реализаций и определения шаблонов целевой среды.

Диалоги редактирования диаграммы классов, диаграмм состояний и диаграмм подключений, диалоги трассировки в отладочном процессе, трансформации построены таким образом, что инженер-проектировщик имеет дело с представлениями, аналогичными изображенным на рисунках 2, 3 и 4.

В четвертой главе приводятся результаты экспериментальных исследований, подтверждающие эффективность предлагаемого подхода. Реализованы различные проектные решения системы управления сверлильной машиной, подсистемы управления гидростанцией в электродуговой плавильной печи, микропроцессорного реле времени и блока регуляции мощности.

Программные реализации для проектных экспериментов в каждом варианте оценивались по 10 параметрам: количество строк кода, затраты памяти программ, затраты памяти данных, суммарное время обработки сигналов, среднее время обработки сигнала, минимальное время реакции на сигнал, максимальное время реакции на сигнал, стандартное отклонение и квадратичное отклонение значений времени реакции на сигнал, сложность ручной модификации.

Оценка параметров порождаемого кода проводилась в сравнении реализаций различных вариантов структурно-функциональной организации целевого кода с реализацией на основе switch-технологии. Показано, что при реализации в табличные представления автоматов (отображение множества переходов в общий список спецификаций и отображение множества переходов в сегментированное множество спецификаций) количество строк кода увеличивается до 42%, объемы памяти программ и данных и время реакции увеличиваются более чем в 20 раз. Однако, данные два варианта отображения имеют самые лучшие значения оценки сложности ручной модификации.

С другой стороны, отображение множества актуальных состояний в множество меток перехода показывает значительный прирост показателей скорости реакции программы на сигнал. Для реализации программы управления сверлильной машиной суммарное, среднее и минимальное время реакции на сигнал уменьшалось более чем на 40%, минимальное время реакции уменьшилось на 27%. При отображении множества актуальных состояний в объекты класса State также замечен прирост в скорости работы программы, но менее значительно (3-6%), зато заметно увеличилась максимальная скорость реакции (около 15%). Схожие результаты получены после применения к диаграмме состояний управления сверлильной машины трансформации «Выделение возвратной части автомата в композитное состояние». А при применении трансформации «Выделение возвратной части автомата в автоматный метод» удалось достичь

сокращения памяти данных на 25% и уменьшения максимального времени реакции на сигнал на 13.5%.

Если сравнивать различные трансформации автоматных диаграмм в рамках одного варианта структурно-функциональной организации порождаемого кода, то можно заметить, что применением трансформации «Выделение возвратной части автомата в автоматный метод» для автомата управления сверлильной машиной удается повысить скорость реакции программы при отображении актуальных состояний в аргумент оператора выбора на 45% и при отображении переходов в общий список спецификаций автомата на 28% относительно автомата без применения трансформаций.

Для автомата управления плавильной печи затраты памяти программ варьируются от –60% до 6% в зависимости от варианта структурно-функциональной организации программы. Значительного прироста скорости получить не удалось (максимум 13% для варианта отображения множества состояний во множество меток переходов). Эффективной оказывается трансформация «Выделение возвратной части автомата в автоматный метод», по нескольким параметрам: до 12% по количеству строк кода, до 10% по объему памяти данных и до 55% сокращения максимального времени реакции. Сравнение различных трансформаций в рамках одного варианта порождения также показывает изменение статических показателей кода (количество строк, затраты памяти программ и данных) в худшую сторону, тогда как скорость реакции программы на сигналы увеличивается даже для обычно «медленных» вариантов отображения множества переходов в общий список переходов и в сегментированное множество спецификаций (до 60% относительно программы без применения трансформаций).

Автомат блока регуляции мощности трансформациям не подвергался в силу простой структуры, однако, при отображении автомата в различные варианты структурно-функциональной организации программ относительно технологии switch получились следующие значения параметров: от –2% до 8% отличия в количестве строк кода, от 5% до 33% отличия в затратах памяти программ, от –113% до 86% отличия в затратах памяти данных.

К автомату микропроцессорного реле времени оказались эффективно применимы трансформации «Инкапсуляция реализации», «Выделение возвратной части автомата в композитное состояние», «Выделение возвратной части автомата в автоматный метод», «Объединение групп состояний с общими спецификациями», комбинация трансформаций «Инкапсуляция реализаций» и «Выделение возвратной части автомата в автоматный метод». Основные параметры кода при выборе различных вариантов генерации без применения трансформаций изменяются в худшую сторону относительно технологии switch. После применения трансформации «Инкапсуляция реализации» скорость работы программы увеличивается на 94% для варианта отображения множества состояний в множество меток перехода. Трансформация «Выделение возвратной части автомата в композитное состояние» заметных приростов параметров кода не дает, а трансформация «Выделение возвратной части автомата в автоматный

метод» увеличивает скорость работы на 94% для варианта отображения множества состояний в объекты класса State. Трансформация «Объединение групп состояний с общими спецификациями» дает прирост скорости реакции до 94%. Комбинация трансформаций выгодна в случае отображения множества состояний в множество меток перехода.

В целом, эксперименты с созданными средствами САПР породили 24 таблицы, обеспечивающие сравнительный анализ проектных решений. Содержимое этих таблиц позволяет сделать вывод, что формируемые в системе проектные решения обладают существенными различиями значений критериальных параметров. Общее число проектных решений, созданных в ходе экспериментирования менее чем за неделю рабочего времени инженера-проектировщика, превосходит 60 (при общем числе операторных строк автоматически сгенерированного кода, превосходящем 67000), что подтверждает высокую эффективность созданных средств автоматизации проектирования.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

1. Разработан подход к автоматизированному проектированию аппаратно-зависимых программных реализаций автоматных диаграмм, который позволяют автоматизировать трудоемкие и рутинные задачи разработки программного обеспечения систем логического управления в части, описываемой диаграммами состояний.
2. Разработанные средства специфицирования аппаратно-зависимых компонентов системы на основе диаграмм подключений позволяют автоматически генерировать спецификации для диаграмм состояний, связанные с аппаратно-зависимой частью системы, что избавляет разработчика от большого объема рутинной работы по определению взаимосвязи микроконтроллера и устройств.
3. Разработаны языки спецификаций автоматных моделей, которые позволяют эффективно разрабатывать модели для конкретного устройства, производить отладку модели на ранних стадиях проектирования и легко переходить от одной целевой платформы к другой.
4. Разработаны и реализованы механизмы структурных и аппаратно-зависимых преобразований автоматных моделей, которые позволяют находить наиболее эффективные решения задач проектирования.
5. Разработаны механизмы и средства визуальной трассировки автоматных диаграмм и интерпретации встроенных операций, позволяющие отладить систему на стадии проектирования и исправить ошибки реализации.
6. Разработаны и реализованы алгоритмы генерации программного кода из автоматных моделей, что позволяет получать максимально эффективные реализации автоматных диаграмм, сокращая время разработки микроконтроллерных систем управления на 30-35%.

ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

Публикации в изданиях, рекомендованных ВАК

1. Антипова Е.В., Негода В.Н. Автоматизация проектирования программно-аппаратных реализаций автоматных диаграмм систем управления // Автоматизация процессов управления. —2012, №1(27). — с. 47-55.

2. Антипова Е.В. Влияние способа преобразования автоматных диаграмм на параметры сгенерированного программного кода // Вестник Волжского университета им. В.Н. Татищева. Серия «Информатика». — 2012,Вып. 20. — с. 111-122.

Публикации в других изданиях

3. Antipova E.V., Negoda V.N. Comparative research of state diagram translation methods // Interactive Systems and Technologies: the Problems of Human-Computer Interaction. Volume III. — Collection of scientific papers.Ulyanovsk: UlSTU, 2009. — 468 p. pp. 177 – 181.

4. Антипова Е.В., Негода В.Н. Использование перегрузки операторов для автоматного программирования // Информатика, моделирование, автоматизация проектирования: сборник научных трудов / под ред. В.Н. Негоды. — Ульяновск : УлГТУ, 2009. — 284 с. С. 46 - 51.

5. Антипова Е.В. Проект генератора исходного кода реализаций диаграмм состояний // Информатика и вычислительная техника : сборнк научных трудов / под науч. ред. П.И. Соснина. — Ульяновск : УлГТУ, 2009. — 275 с. С. 10 – 15.

6. Антипова Е.В. Разработка структуры модели генерации исходного кода на основе автоматных диаграмм // Всероссийская конференция с элементами научной школы для молодежи «Проведение научных исследований в области обработки, хранения, передачи и защиты информации», 1-5 декабря 2009 г. Россия, Ульяновск : сборник научных трудов. В 4 т. Т. 3. — Ульяновск, 2009. — 419 с. С. 198 – 205.

7. Антипова Е.В. Среда быстрого проектирования систем логического управления (на основе диаграммных моделей) // Информатика, моделирование, автоматизация проектирования: сборник научных трудов / под ред. В.Н. Негоды. — Ульяновск : УлГТУ, 2010. — 640 с. С. 3 – 8.

8. Антипова Е.В. Формальная модель сквозных переходов UML // Информатика и вычислительная техника : сборник научных трудов / под ред. Н.Н. Войта. — Ульяновск : УлГТУ, 2011. — 656 с. С. 37 – 42.

9. Антипова Е. В., Негода В.Н. Разработка протокола взаимодействия автоматной модели и целевой платформы в процессе имитационного тестирования автоматных диаграмм // Информатика, моделирование, автоматизация проектирования: сборник научных трудов / под ред. Н.Н. Войта. — Ульяновск : УлГТУ, 2011. — 416 с. С. 39 – 44.

10. Antipova E., Negoda V. Using automatic methods for automata models development // Interactive Systems and Technologies: the Problems of Human-Computer Interaction. – Collection of scientific papers. — Ulyanovsk:UlSTU, 2011. — 435 p. pp. 104 – 109.

Подписано в печать 21.11.2012. Формат 60x84/16.

Усл. печ. л. 1,16. Тираж 100 экз. Заказ № 1027.

Типография УлГТУ. 432027. Ульяновск, Сев. Венец, 32.