

На правах рукописи

Кубасов Сергей Валерьевич

**Верификация автоматных программ в
контексте синхронного программирования**

05.13.11 – Математическое и программное обеспечение вычислительных
машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ

диссертации на соискание ученой степени
кандидата технических наук

Ярославль – 2008

Работа выполнена на кафедре теоретической информатики
Ярославского государственного университета им. П.Г. Демидова.

Научный руководитель: *доктор физико-математических наук,
профессор
Соколов Валерий Анатольевич*

Официальные оппоненты: *доктор физико-математических наук,
профессор
Ломазова Ирина Александровна;
кандидат физико-математических наук,
доцент
Дехтярь Михаил Иосифович*

Ведущая организация: *Санкт-Петербургский государственный
университет информационных техноло-
гий, механики и оптики*

Защита состоится 14 ноября 2008 г. в 14 часов на заседании диссертационного совета ДМ002.084.01 при Учреждении Российской академии наук Институте программных систем РАН, расположенном по адресу: 152020, Ярославская область, Переславский район, с. Веськово, ул. Петра Первого, д. 4а.

С диссертацией можно ознакомиться в библиотеке Учреждения Российской академии наук Института программных систем РАН.

Автореферат разослан «___» октября 2008 г.

Ученый секретарь
диссертационного совета

Пономарева С.М.

Общая характеристика работы

Актуальность работы С 90-х годов XX века в России развивается автоматное программирование. Профессор А.А. Шалыто предложил использовать switch-технологию (другое название автоматного программирования) для решения задач логического управления. Ее основная идея в использовании автоматов для кодирования логики программы. Допускается применять другие подходы для решения отдельных подзадач. Технология автоматного программирования значительно упрощает взаимодействие различных участников процесса разработки программного обеспечения.

Со времени своего изобретения технология претерпела некоторые изменения. Были предложены различные модификации switch-технологии, использующие идеи других популярных парадигм программирования, например, объектно-ориентированного программирования. Расширилась область применения. В результате появилось несколько вариаций АП, среди которых уже сложно выявить главное направление.

Для разработки сложных программных систем с применением switch-технологии был создан проект UniMod. В этом инструментальном средстве можно было обнаружить множество полезных функций для разработчика. Это визуальное программирование, отладчик, подсветка синтаксиса и многое другое. К сожалению, не было никакой возможности верификации программ. Более того, заложенные в основу представления об автоматной программе делали задачу добавления полноценной проверки проблематичной. UniMod не дает четкого определения автоматной программе, тем более не использует математическую модель.

Все эти соображения привели к необходимости разработки новой среды программирования и верификации с упором на функцию проверки. Основная трудность в осуществлении этого замысла заключалась в выборе методов

верификации автоматных программ. Хотя задача верификации уже долгое время служит предметом пристального внимания со стороны многих ученых, ее нельзя назвать решенной, верификация же автоматных программ является новым направлением.

Задачи логического управления часто предъявляют критические требования к надежности программного обеспечения. Цифровые устройства сейчас можно встретить практически во всех сферах человеческой жизни. От их бесперебойного функционирования часто зависит жизнь людей. Со временем количество и сложность устройств логического управления только возрастают. Поэтому задача верификации не теряет, а, наоборот, усиливает свою актуальность. Ручная проверка — трудоемкое занятие. Гораздо удобнее и надежнее поручить выполнение этой задачи компьютерной программе.

Задача верификации возникла несколько десятилетий назад. К настоящему моменту уже выработаны разнообразные подходы ее решения. Среди ученых, внесших значительный вклад в решение этой задачи, можно выделить Н.А. Анисимова, О.Л. Бандман, Ю.Г. Карпова, И.А. Ломазову, В.А. Непомнящего, А.К. Петренко, Р.Л. Смелянского, В.А. Соколова, P.A. Abdulla, G. Berry, M.C. Browne, K. Cerans, E.M. Clarke, E.A. Emerson, A. Finkel, Jr.O. Grumberg, G.L. Holzmann, K. Jensen, Z. Manna, A. Pnueli, Ph. Schnoebelen, N. Sidorova, J. Sifakis.

Накопленный опыт по верификации разнообразных систем может быть применен для автоматного программирования. У каждого подхода верификации есть свои плюсы и минусы, однако для АП наиболее подходящим методом является проверка на моделях. Автоматная программа уже состоит из автоматов, формальных по своей природе. Достаточно только доопределить, уточнить правила использования и взаимодействия автоматов, чтобы получилась модель, приемлемая для верификации.

Задача верификации автоматных программ находится в процессе иссле-

дования. Можно выделить две группы, активно работающих над этой проблемой. В Ярославском государственном университете им. П.Г. Демидова на кафедре теоретической информатики проводятся исследования по моделированию, спецификации и верификации автоматных программ. Результаты работ обсуждаются на семинаре “Моделирование и анализ информационных систем” и конференциях. В работу вовлекаются студенты. Команда Шалыто А.А., автора технологии автоматного программирования, также достигла определенных успехов в разработке среды для верификации автоматных программ.

В работе Кузьмина Е.В.¹ предлагается иерархическая модель автоматных программ. Модель рассматривается на примере системы управления кофеваркой. В работе Васильевой К.А. и Кузьмина Е.В.² предлагается способ моделирования, спецификации и верификации автоматных программ. Используется верификатор SPIN, логика LTL. Рассмотрен пример системы управления банкоматом. Одним из развитий иерархической модели автоматных программ является модель, использующая формализм сетей Петри. Разработка программы выполняется в UniMod, применяется верификатор CPN/Tools, рассмотрен пример системы управления кофеваркой. Получено свидетельство об официальной регистрации программы для ЭВМ.

На кафедре технологий программирования СПбГУ ИТМО были получены в том числе следующие результаты. В работе Вельдера С.Э.³ рассматривается техника верификации автоматных программ, состоящих из одного конечного автомата. Описывается преобразование автомата в структуру Крип-

¹Кузьмин, Е.В. Иерархическая модель автоматных программ // Моделирование и анализ информационных систем. — 2006. — Т.13, № 1. — С. 27–34.

²Васильева, К.А. Верификация автоматных программ с использованием LTL / К.А. Васильева, Е.В. Кузьмин // Моделирование и анализ информационных систем. — 2007. — Т.14, № 1. — С. 3–14.

³Вельдер, С.Э. О верификации простых автоматных программ на основе метода “Model Checking” / С.Э. Вельдер, А.А. Шалыто // Информационно-управляющие системы. — 2007. — № 3. — С. 27–38.

ке, выражение темпоральных свойств на языке СТЛ, верификация по модели (метод “Modal Checking”) и построение сценария для исходного автомата, если был найден контрпример. Техника верификации демонстрируется на примере универсального инфракрасного пульта для бытовой техники. Применение метода Model Checking также исследуется в других работах.

Можно заметить, что все авторы используют Model Checking. Главное различие наблюдается в моделях. Данная работа не является исключением.

Язык синхронного программирования `esterel` разрабатывался для аналогичного круга задач, что и автоматное программирование. В отличие от АП, в `esterel` был изначально заложен прочный теоретический фундамент. В `esterel` существует базис — подмножество языка, на котором можно определить все его операторы. Программа на языке `esterel` — это уже модель, готовая для верификации. Существует верификатор `Xeve` для проверки `esterel` программ. АП и язык `esterel` помимо общей задачи роднит тот факт, что оба они используют бинарные сигналы.

АП и `esterel` удачно дополняют друг друга. АП предлагает средства для визуальной разработки программ, а `esterel` обеспечивает прочную математическую основу и средства верификации.

`Xeve` — это верификатор для `esterel` программ, представляемых в виде конечных автоматов. `Xeve` предлагает дружелюбный графический пользовательский интерфейс.

Цель диссертационной работы Главная цель диссертации — создать программный комплекс разработки и верификация автоматных программ. Достижение указанной цели было связано с решением следующих подзадач.

- Разработать формальную модель автоматной программы, которая бы подходила для сложившейся практики применения автоматного программирования.

- Разработать метод верификации автоматных программ.
- Разработать и реализовать программные средства построения и верификации автоматных программ. Исследовать работу программной системы при решении практических задач.

Научная новизна Все основные результаты являются новыми.

- Разработана формальная модель автоматной программы, учитывающая наиболее важные идеи автоматного программирования.
- К автоматному программированию был применен синхронный подход. Уточнена временная модель. Поведение программы стало детерминированным. Разработана вариация автоматного программирования — синхронно-автоматное программирование.
- Разработаны способы верификации синхронно-автоматных программ при помощи существующих программных инструментов языка `esterel`. Использован верификатор `Xeve`.
- Создана программная среда разработки и верификации синхронно-автоматных программ.

Практическая ценность Разработаны методы верификации автоматных программ. Их применение позволит обнаружить многие ошибки, допускаемые в процессе разработки. Возможна проверка программы на соответствие изначальным требованиям технического задания.

Разработана и реализована программная система разработки и верификации синхронно-автоматных программ. Ее применение упрощает и ускоряет разработку и проверку указанного класса программ.

Апробация работы Результаты диссертации докладывались на 7-ой международной конференции и выставке “Системы проектирования, технологической подготовки производства и управления этапами жизненного цикла промышленного продукта (CAD/CAM/PDM-2007)” (Москва, 2007), XIV-ой международной научно-практической конференции “Современные техника и технологии” (Томск, 2008), международной научной конференции “Информация, сигналы, системы: вопросы методологии, анализа и синтеза” (Таганрог, 2008), международной научной конференции “Математика, кибернетика, информатика” (Ярославль, 2008).

Результаты обсуждались на семинаре “Моделирование и анализ информационных систем” кафедры теоретической информатики Ярославского государственного университета им. П.Г. Демидова (2006–2008 гг.).

Публикации По теме диссертации опубликовано 7 научных работ. Из них 3 опубликованы в изданиях, входивших в перечень ВАК на момент публикации и находящихся в перечне ВАК в настоящий момент.

Личный вклад автора Все результаты исследований, составляющих основное содержание диссертации, получены автором самостоятельно.

Структура и объем диссертации Диссертация состоит из введения, трех глав и заключения. Объем работы составляет 122 страницы в формате машинописного текста. Список литературы содержит 88 наименований.

Краткое содержание работы

Во введении обоснована актуальность диссертационной работы, сформулирована цель и аргументирована научная новизна исследований, показана практическая значимость полученных результатов, представлены выносимые на защиту научные положения.

В первой главе описываются основы автоматного программирования,

исследуются подходы к верификации автоматных программ, обосновывается выбор синхронного подхода для проверки программ.

Автоматное программирование было разработано для решения задач логического управления. По мнению профессора Шалыто, существующие подходы к разработке указанных систем обладают теми или иными недостатками. В первую очередь, они неудобны в использовании. Затруднено взаимодействие различных участников процесса разработки. В результате была предложена технология программирования, в основе которой лежат взаимодействующие автоматы.

Идея применения автоматов была развита в многочисленных работах его последователей. Предлагались различные варианты размещения внешних вызовов в автоматах. В одних случаях внешние воздействия можно вызывать только на переходах, в других — разрешаются воздействия при входе и выходе из состояния. Несколько состояний группируются в одно состояние. В результате появляется иерархия состояний по вложенности. Сравнительно сложная программа включает в себя несколько автоматов, поэтому важны способы их взаимодействия. Предлагались библиотеки, поддерживающие автоматную разработку программ. Реализованы вспомогательные утилиты. Следует отметить UniMod — среду разработки автоматных программ на языке Java. Она оказала важное влияние на данную работу.

Программисту дается большая свобода в выборе способов взаимодействия автоматов. В большинстве случаев невозможно проследить структуру программы. Автоматы используются в отдельных местах кода программы, разработанной в соответствии с одной из популярных технологий программирования (объектно-ориентированной, структурной). Автоматы взаимодействуют опосредованно через код.

В формальной модели мы не можем допустить подобные вольности. Для того, чтобы получившаяся модель позволяла проверять полезные свойства,

необходимо провести четкую границу между автоматным и неавтоматным кодом. Их взаимодействие должно выполняться по строго определенным правилам. Модель будет охватывать только автоматную часть.

Подобные требования приводят к необходимости разработки специального варианта автоматной технологии, в котором автоматная часть программы имеет четкие границы. Из всего разнообразия предложений было решено взять за основу первую книгу, посвященную switch-технологии⁴.

В ней достаточно подробно описываются основные элементы автоматной программы, и убедительно обосновывается, почему было выбрано то или иное решение. Становится понятна изначальная задумка switch-технологии. Важной особенностью автоматного подхода, излагаемого в этой книге, является условие, что все сигналы бинарные. С одной стороны, это значительное ограничение, с другой стороны, оно ближе по духу задаче логического управления.

Система взаимодействующих автоматов по своей природе близка к формальной модели. Достаточно небольших изменений, чтобы автоматная программа стала полноценной формальной моделью. Для верификации подходит метод Model Checking. Тому есть несколько причин. Во-первых, как уже упоминалось, автоматная природа модели. Во-вторых, возможность полностью автоматической проверки.

Удачной находкой для задачи верификации автоматной программы явилось синхронное программирование и язык esterel. Язык esterel, как и автоматное программирование, разрабатывался для решения задач логического управления. В настоящее время это зрелое решение, активно применяющееся на практике.

Для синхронного программирования характерна особая временная мо-

⁴Шалыто, А.А. Switch-технология. Алгоритмизация и программирование задач логического управления. — СПб.: Наука, 1998. — 628 с.

дель. Время протекает как последовательность дискретных инстантов. Каждый инстент программа выполняет вычисления. Считается, что длительность инстента равна нулю, все вычисления выполняются мгновенно. Задача программы в каждый инстент — вычислить значения выходных сигналов и следующее состояние системы в зависимости от текущего состояния и значений входных сигналов. Поведение программы детерминировано. Т.е. для любого допустимого состояния и любого допустимого набора входных сигналов следующее состояние и значения выходных сигналов должны определяться однозначно. Порядок вычислений в пределах инстента не имеет значения. Важен только результат. Такой подход позволяет значительно сократить количество состояний, возникающих в процессе верификации. Значит, верификацию можно выполнить для систем большего размера.

Для автоматного программирования вопрос о временной модели не имеет четкого ответа. АП применяется для реактивных, интерактивных, а в некоторых случаях, для практически любых систем. Обычно решение о временной модели оставляется программисту. Синхронный подход ограничивает нас использованием реактивных систем, что закономерно для класса задач логического управления.

Верификатор Xeve используется для проверки esterel программ. Esterel программа представляется в виде набора конечных автоматов. Описание сохраняется в формате BLIF (Berkeley Logical Interchange Format). Перед верификацией строится пространство состояний программы в виде BDD (Binary Decision Diagrams). Инструмент Xeve может выполнять две функции: минимизация автомата и проверка статуса выходных сигналов. Вторая функция используется для верификации. Можно проверить два типа утверждений: 1) выходной сигнал никогда не генерируется, 2) выходной сигнал генерируется в каждом инстенте.

В общем случае хотелось бы иметь возможность проверять более слож-

ные свойства. Проверяемое свойство выражается отдельной программой-наблюдателем (observer) на языке estereel, которая подсоединяется к основной программе. При нарушении свойства наблюдатель генерирует специальный сигнал. Таким образом, выполнение свойства эквивалентно утверждению, что тревожный сигнал никогда не генерируется. Существует приложение, позволяющее описывать проверяемые свойства на языке LTL (Linear Time Temporal Logic).

Во второй главе описывается среда разработки и верификации синхронно-автоматных программ. Первый раздел посвящен теоретическим аспектам, лежащим в основе. Затем идет язык описания автоматной программы, верифицируемые свойства, структура среды разработки.

Объединение автоматного программирования и синхронного подхода привело к созданию вариации АП, названной синхронно-автоматным программированием.

Модель программы состоит из двух типов модулей: автоматов и синхронных сетей. Автомат $A = (Q, q_1, X, Y, \Phi)$ состоит из множества состояний Q , начального состояния q_1 , множества входных X и выходных Y сигналов, функции переходов Φ . Функция переходов $\Phi: Q \times L \rightarrow Q \times R$ определяет условие перехода $l \in L$, список генерируемых сигналов $r \in R$ и новое состояние. Условие перехода представляет из себя логическую формулу над значениями входных сигналов: $L ::= 0 \mid 1 \mid x_i \mid L \wedge L \mid L \vee L \mid \neg L \mid (L)$. Выходные сигналы могут генерироваться только на переходах. Допускаются петли (исходное и конечное состояния совпадают).

Синхронная сеть $N = (\mathcal{I}, X, Y, G)$ используются для группировки подсистем. Подсистемами являются автоматы и синхронные сети. Сначала синхронные сети создаются только из автоматов, затем уже созданные сети могут быть использованы при построении других сетей. Все, что синхронная сеть делает, выходные сигналы одних подсистем с входными сигналами дру-

гих. \mathcal{I} — набор синхронных подсистем, использованных для построения данной синхронной сети. Синхронная сеть сама обладает интерфейсными сигналами. X и Y — множество входных и выходных сигналов соответственно. Интерфейсные сигналы сети используются наравне с сигналами ее компонентов. G — отношение смежности на множестве сигналов, определяющее группы связанных сигналов.

Синхронно-автоматная программа преобразуется в программу на языке `esterel`. Отдельные модули исходной модели преобразуются в аналогичные модули языка `esterel`. Сохраняются имена компонентов.

С-А программу можно описать текстуально в формате XML. Однако такое представление крайне неудобно для разработки программы. Поэтому была создана система визуальных обозначений на основе языка UML. Отображение компонентов программы на элементы UML соответствует интуитивным ожиданиям. Автоматы моделируются с помощью диаграмм автоматов, состояния соответствуют состояниям UML. Метки на дугах имеют специальный формат. Синхронные сети изображаются с помощью диаграмм композитной структуры (Composite Structure Diagram). Сигналы автоматов и синхронных систем моделируются портами.

Структуру среды разработки удобно представить с помощью двух диаграмм. На рисунке 1 показаны пути преобразования данных в процессе работы. Рисунок 2 содержит соответствующие инструменты, используемые для создания или преобразования данных с рис. 1. Каждой стрелке, символизирующей операцию преобразования данных, на рис. 1 соответствует прямоугольник инструмента на рис.2.

Область рисунка, окруженная пунктирной рамкой, является опциональной. Так, на рис. 1 компонент «Проверяемые свойства LTL» является необязательным. Проверяемые свойства можно определять сразу на языке `esterel` (прямоугольник «Проверяемые свойства `esterel`»).

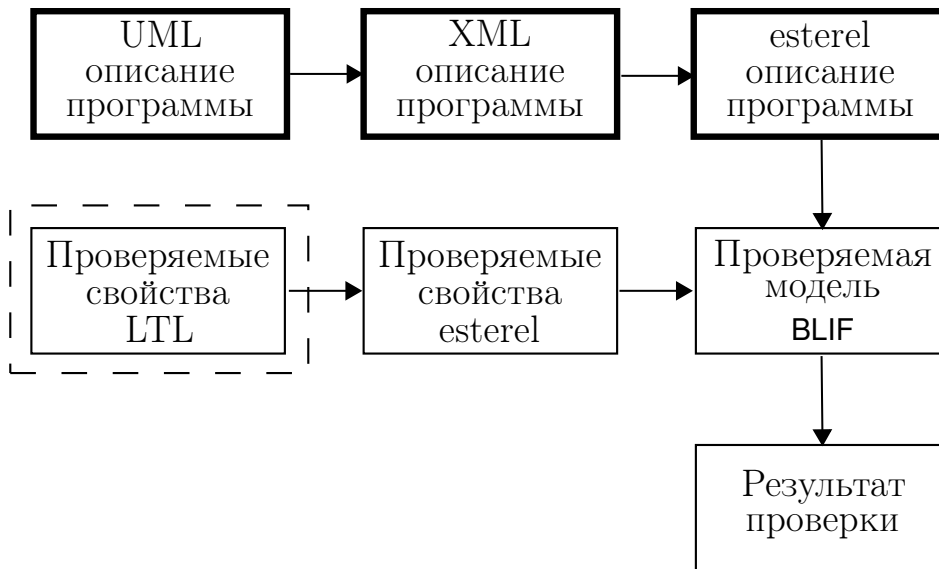


Рис. 1. Среда разработки синхронно-автоматных программ. Диаграмма данных

Прямоугольники с жирной рамкой отмечают компоненты, разработанные автором. Все остальные компоненты, соответственно, были заимствованы.

Компилятор языка *estereel*, верификатор *Xeve* и другие инструменты, обеспечивающие основную функцию верификации, спроектированы для работы на нескольких платформах. Дистрибутив компилятора предлагается для Windows NT, Linux, Sun Solaris, IBM AIX, Dec OSF1. *Xeve* поддерживает платформы: Linux, Sun Solaris, Dec Alpha OSF. В связи с этим при разработке программной среды большое внимание уделалось переносимости. *MagicDraw UML*, выбранный в качестве UML редактора, работает на Java платформе. Скрипты *MDUML2SAM*, *SAM2str1* написаны на Perl.

Разработка программы выполняется в среде неспециализированного UML редактора. Используются стандартные возможности редактора, причем далеко не все. Различные дополнительные инструменты программы упрощают разработку. Очень удобна возможность комментировать практически любой элемент модели. Написано несколько вспомогательных утилит, автоматизирующие процесс преобразования описания С-А программы из набора UML диаграмм в XML описание, а затем в код на языке *estereel*.

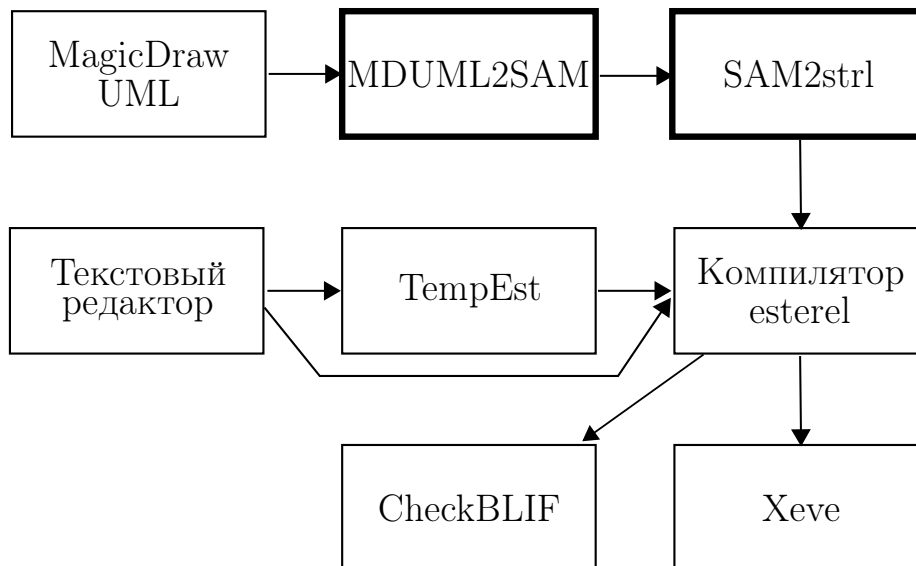


Рис. 2. Среда разработки синхронно-автоматных программ. Диаграмма инструментов

Верификация С-А программы выполняется инструментом Xeve. Проверяемые свойства должны быть описаны на языке esterel в виде отдельных модулей. Взаимодействие с основной программой (полученной ранее), как правило, не представляет затруднений, т.к. структура esterel программы в основном повторяет структуру С-А модели, хотя знание языка esterel определенно является необходимым. Основная программа объединяется с модулями проверки. Далее процедура верификации совпадает с той, которая применяется для чистого языка esterel. Процедура в большей части автоматизирована. В случае получения отрицательного ответа (свойство нарушено), необходимо проинтерпретировать трассу, приводящую к ошибке. Отдельно стоит отметить инструмент TempEst, служащий надстройкой процесса верификации. С его помощью свойства можно задавать на языке LTL.

В третьей главе описываются результаты применения предлагаемой технологии синхронно-автоматного программирования для решения практических задач.

Первый пример — арбитры шины. Задача арбитра шины заключается в управлении доступом к разделяемому ресурсу. Только один из пользователей

имеет возможность получить доступ к шине в каждый момент времени. Необходимо поддерживать справедливость в предоставлении доступа к ресурсу. Пользователь, непрерывно запрашивающий ресурс, должен не позднее чем через заданное время получить доступ.

Решение этой задачи очень специфично для синхронного программирования. Тем не менее, она была успешно реализована синхронно-автоматным способом. Используются четыре автомата и две синхронных сети. Решение наглядно представляется при помощи диаграмм UML. Были проверены все важные свойства.

Второй пример моделирует работу часов с будильником. Часы-будильник обладают такими возможностями: отображение текущего времени, установка времени звонка, выбор мелодии звонка, включение/выключение звонка, установка/отмена режима ежечасных оповещений.

Реализация включает в себя четыре автомата и одну синхронную сеть. Было решено, что функция отсчета времени не входит в обязанности программы. Синхронно-автоматная программа не имеет переменных для хранения многозначных величин таких, как текущее время. Моделировать целочисленный регистр при помощи бинарных сигналов возможно, но крайне неудобно. С-А программа не предназначена для этого. Мы предполагаем, что существуют внешние регистры, в которых хранятся реальные значения времени. Программа выполняет инкрементирование, декрементирование, сравнение с нулем этих счетчиков.

Проверяемые свойства представляют собой утверждения, что определенные комбинации сигналов несовместимы, состояния автоматов, значения сигналов, находятся в определенном отношении. В процессе разработки программы делается множество допущений, выполнение которых неочевидно после реализации. Функция верификации позволяет проверить такие допущения.

Третий пример моделирует работу микроволновой печи. В ней преду-

смотрены такие функции: выбор времени приготовления, паузы, разморозки; установка текущего времени; отображение и отсчет текущего времени; выбор уровня мощности микроволнового генератора. Приготовление заключается в прохождении последовательности фаз. Процесс может быть приостановлен в любой момент. Предусмотрена функция автоматического отключения микроволнового генератора при открытии дверцы печи. После закрытия дверцы приготовление будет возобновлено.

Это наиболее сложный из всех реализованных примеров. В нем использовано 9 автоматов и одна синхронная сеть. Кроме того, для проверки некоторых свойств были построены дополнительные элементы.

Программа предполагает существование специальных внешних объектов таких, как таймеры и регистры. Некоторые функции удобнее реализовать внешними устройствами.

Проверяемые свойства, главным образом, относятся к деталям реализации программы. Чем сложнее программы, тем больше деталей, за которыми трудно уследить. Разработка программы сопровождалась контролем при помощи верификатора. Описано более двух десятков свойств. Были обнаружены и исправлены ошибки. Таким образом, можно утверждать, что функция верификации способствует разработке качественной программы.

В заключении подводятся итоги применения синхронно-автоматного подхода для решения задачи верификации автоматных программ.

Основные результаты и выводы

В рамках диссертации получены следующие результаты.

- Разработана формальная модель автоматной программы, основанная на синхронном подходе. Получившаяся вариация автоматного программирования была названа синхронно-автоматным программированием.

- Разработан метод верификации синхронно-автоматных программ с использованием верификатора Xeve.
- Разработана графическая нотация описания и XML формат хранения синхронно-автоматной программы. Создана программная среда разработки и верификации синхронно-автоматных программ.

Публикации по теме диссертации

1. Кубасов, С.В. Синхронная модель автоматной программы / С.В. Кубасов, В.А. Соколов // Моделирование и анализ информационных систем. — 2007. — Т.14, № 1. — С. 11–18.
2. Кубасов, С.В. Верификация синхронно-автоматных программ // Моделирование и анализ информационных систем. — 2007. — Т.14, № 4. — С. 20–27.
3. Кубасов, С.В. Система разработки синхронно-автоматных программ для решения задач логического управления // Материалы международной конференции и выставки CAD/CAM/PDM-2007. — М.: Институт проблем управления РАН, 2007. — С. 53–54.
4. Кубасов, С.В. Синхронно-автоматное программирование для задач логического управления // Вопросы современной науки и практики. — Тамбов: Тамбовский государственный технический университет, 2007. — Т.2, № 4(10). — С. 230–233.
5. Кубасов, С.В. Разработка и верификация синхронно-автоматных программ на примере микроволновой печи // Труды XIV международной научно-практической конференции “Современные техника и техноло-

гии”. — Томск: Томский политехнический университет, 2008. — Т. 2. — С. 324–325.

6. Кубасов, С.В. Разработка и верификация автоматных программ средствами синхронного программирования // Материалы международной научной конференции “Информация, сигналы, системы: вопросы методологии, анализа и синтеза”. — Таганрог: Изд-во ГТИ ЮФУ, 2008. — Т. 5: Радиоэлектронные системы и устройства. — С. 26–32.
7. Кубасов, С.В. Верификация синхронно-автоматных программ с использованием LTL // Моделирование и анализ информационных систем. — 2008. — Т.15, № 2. — С. 46–49.

В работе [1] автором разработана синхронная модель автоматной программы. Все остальные работы были выполнены без соавторов.