УДК 004.312.466

С. Н. ХАРЛАП, кандидат технических наук, Б. В. СИВКО, магистр технических наук, Белорусский государственный университет транспорта, г. Гомель

ВЕРИФИКАЦИЯ ЦИКЛИЧЕСКИХ СИСТЕМ ЖЕЛЕЗНОДОРОЖНОЙ АВТОМАТИКИ И ТЕЛЕМЕХАНИКИ

Рассмотрены особенности верификации железнодорожных микропроцессорных циклических систем реального времени с помощью формальных методов. Обосновано разделение доказательства корректности на инициализацию и циклическое выполнение для упрощения верификации и возможности перехода к проверке модели. Сформулированы необходимые условия, подлежащие доказательству посредством дедуктивного анализа. Рассмотрены способы верификации, позволяющие проверять заданные на этапе валидации свойства или построить модель на основе исходного кода программы.

Введение. Одной из задач построения безопасных микроэлектронных систем железнодорожной автоматики и телемеханики (СЖАТ) является разработка и верификация безопасных аппаратно-программных комплексов (АПК) реального времени, ответственных за управление технологическими процессами, связанными с безопасностью движения поездов. К таким системам относятся микропроцессорная автоблокировка АБТЦ-М, микропроцессорная централизация стрелок и сигналов «Ипуть», система счета осей ЭССО, элементами которых являются самостоятельные АПК, выполняющие определенные функции, такие, как безопасное управление напольными объектами, контроль рельсовых цепей, управление светофорами и т. д.

Большинство СЖАТ относится к реактивным системам (*reactive systems*) реального времени, которые работают бесконечно долго и их поведение определяется внешними сигналами и внутренним состоянием [1, 2]. Кроме того, железнодорожные АПК имеют свои особенности исходя из специфики предметной области, и в связи этим данные системы выполняют поставленные перед ними задачи преимущественно в циклическом режиме (*cyclic executive*), который наиболее подходит для проектирования безопасных систем [3].

Программное обеспечение (ПО) рассматриваемых АПК является неотъемлемым компонентом и подлежит обязательному анализу на безопасность. В настоящее время отсутствуют универсальные решения для верификации ПО, и идет интенсивный поиск методов и средств, позволяющих эффективно проводить мероприятия по обеспечению необходимого уровня качества ПО и последующему доказательству безопасности.

В лаборатории «Безопасность и ЭМС технических средств» Белорусского государственного университета транспорта проводятся испытания устройств на безопасность их функционирования, одним из этапов которого является поиск ошибок в ПО, который включает в себя доказательство корректности с помощью формальных методов (*formal methods*) [4, 5]. СЖАТ относятся к критически важным объектам информатизации, для которых стандарт IEC 61508 имеет градацию уровней полноты безопасности от SIL-1 до SIL-4. Железнодорожные системы должны соответствовать наиболее строгому уровню SIL-4, который настоятельно рекомендует применение формальных методов для критически важных систем информатизации [6].

В статье рассматривается доказательство корректности уже определенной функции безопасности (ФБ) для ПО реактивных систем реального времени, работающих в циклическом режиме [7].

1 Этапы инициализации и циклического выполнения. Реактивные системы взаимодействуют с окружением посредством передачи и приема сигналов, как показано на рисунке 1.

Рисунок 1 – Реактивная система и её взаимодействие с окружением

Такие системы работают бесконечно долго и их поведение определяется внутренним состоянием.

Основной причиной того, что большинство железнодорожных АПК проектируются на основе архитектуры циклического выполнения (*cyclic executive*), является то, что системы данного типа более контролируемы и просты алгоритмически, а это упрощает их верификацию. При этом специфика предметной области такова, что не требуется быстрая реакция на изменение внешних параметров.

Для СЖАТ, обладающих описанными свойствами, удобно выполнить декомпозицию доказательства безопасности на два различных этапа работы (рисунок 2):

– инициализация системы;

– циклическое выполнение.

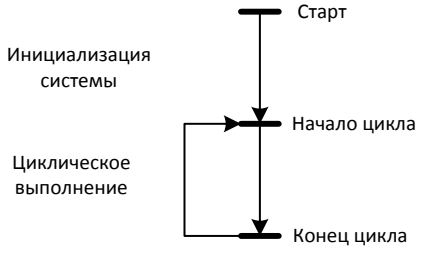

Рисунок 2 – Инициализация и циклическое выполнение

Первый этап выполняется один раз и обеспечивает выход системы в некоторое начальное состояние. В

дальнейшем система работает в некотором цикле, который повторяется бесконечно долго.

С точки зрения верификации система должна всегда удовлетворять одной и той же ФБ во все время функционирования, но особенности верификации рассматриваемых этапов различны. Циклическое выполнение происходит бесконечно, и все это время АПК обязан выполнять некоторый инвариант I, зависящий от текущего и предыдущих состояний системы x и являющийся формализованным условием $I(x)$. Инициализация происходит один раз, и её задачей является обеспечение корректного состояния инварианта на начало цикла.

В связи с качественным отличием этапов подходы их доказательства безопасности различны. Инициализация системы является алгоритмом, имеющим вход и выход, где применяются традиционные методы верификации [8, 9]. Во время цикла вычисления никогда не заканчиваются, а анализ системы направлен на её состояния и переходы между ними. Здесь необходимо либо доказывать истинность инварианта посредством дедуктивного анализа, либо осуществлять переход к модели и её проверке [10, 11].

2 Способы верификации циклических СЖАТ. Будем считать, что рассматриваемая система последовательно, на начало каждого i-го витка цикла, находится в состояниях S_0, S_1, …, S_i. Иллюстрация изменения состояний циклической системы показана на рисунке 3.

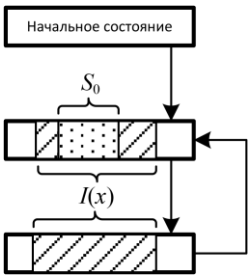

Рисунок 3 – Доказательство инварианта циклической системы

Изначально система стартует в некотором состоянии, определяемом характеристиками микроконтроллера и его окружением. Во время инициализации происходит проверка и настройка параметров таким образом, чтобы была возможна корректная работа в циклическом режиме, тем самым АПК выходит в одно из начальных состояний цикла множества S_0. Данный этап верифицируется с помощью традиционных методов доказательства корректности, когда происходит получение постусловия S_0 из начального состояния. Это может быть вывод как в прямом порядке (определение всех точек достижения из начального состояния), так и в обратном (вывод начального состояния из точки начала цикла и, соответственно, определение множества S_0, при котором это возможно).

Для доказательства корректности необходимо доказать, что:

– система после инициализации всегда выполняет $I(x)$;

– $I(x)$ является истинным на каждом витке цикла.

То, что система после инициализации всегда выполняет $I(x)$, проверяется посредством выражения

$$S_0(x) \rightarrow I(x). \quad (1)$$

Во время проверки истинности $I(x)$ на каждом витке цикла для облегчения доказательства корректности может быть выбран другой инвариант $I_p(x)$, который должен быть таким же или более строгим, чем $I(x)$, то есть чтобы выполнялось условие

$$I_p(x) \rightarrow I(x). \quad (2)$$

Это делается потому, что значение $I(x)$ не всегда является простым или очевидным выражением, и об этой проблеме говорит опыт многих коллективов [12]. Поиск условия $I(x)$ может занимать больше времени, чем его проверка, так как выбранный инвариант оказывается слишком слабым и не выполняет требуемых условий или оказывается слишком сильным и доказать его истинность не является возможным.

Для выполнения проверки истинности $I(x)$ на каждом витке цикла могут быть использованы два следующих подхода, каждый из которых имеет свои особенности и ограничения:

– доказательство $I(x)$ на каждом витке цикла посредством дедуктивного анализа;

– последовательный вывод всех возможных состояний, для каждого из которых проверяется истинность $I(x)$.

3 Верификация циклических СЖАТ доказательством корректности истинности инварианта. Данный подход требует вывода истинности $I(x)$ в зависимости от предыдущего или последующего состояния S_i. Другими словами, необходимо доказать, что если система удовлетворяет $I(x)$ на начало цикла, то из этого следует, что система будет выполнять $I(x)$ на последующем начале цикла. Таким образом, если система достигла корректного состояния, то требуется доказать, что она сохранит корректность состояния на следующем цикле и, по индукции, будет выполнять $I(x)$ всегда.

Процедура доказательства проводится посредством дедуктивного анализа и поэтому не может быть автоматизирована. Однако для небольших систем или для случаев с малым числом условий и состояний данный подход может позволить быстро и эффективно провести верификацию.

4 Верификация циклических СЖАТ с помощью последовательного вывода состояний. Если было получено множество начальных состояний S_0, то можно из него получить множество состояний, которые может принимать система на начало следующего цикла S_1. Как правило, после проведения такой операции множество S_1 оказывается больше, чем S_0. В дальнейшем данную операцию можно повторить несколько раз, что показано на рисунке 4.

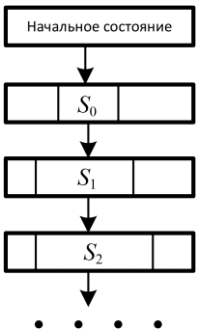

Рисунок 4 – Последовательный вывод возможных состояний, которые принимает система

Некоторое малое число итераций может позволить сформулировать зависимости, присутствующие в переходах системы из одного состояния в другое, а также сформировать инвариант $I(x)$. Таким образом, если будет получена пара одинаковых S_i, S_{i+n}, то это означает, что АПК будет повторять свои состояния, что и является тем инвариантом, который выполняет система. А если были определены зависимости, то на их основании можно сформулировать общий инвариант системы и в дальнейшем его проверить.

На основе исходного кода посредством доказательства свойств системы можно построить правило $R(S_i, S_{i+1})$, описывающее переход системы из одного состояния в другое, что показано на рисунке 5.

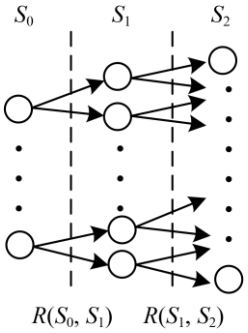

Рисунок 5 – Графический вывод возможных состояний

Наличие данного правила, конечного числа состояний системы и множества состояний S_0 позволяет перейти к проверке модели, например к построению сети Петри или модели Крипке [11]. Это дает возможность как верификации на другом уровне абстракции, так и проведения автоматической верификации.

5 Разработка и верификация самостабилизирующихся систем. Выполнение условий выхода на корректное начальное состояние и его сохранение в течение цикла позволяет разрабатывать и верифицировать системы, обладающие свойством самостабилизации (*self-stabilization systems*) [13, 14].

СЖАТ являются распределенными системами, устройства которых работают независимо и обеспечивают некоторые свои внутренние условия безопасности, на основании которых выполняется глобальный инвариант, являющийся требованием к безопасности всего АПК. Например, микропроцессорная централизация «Ипуть» должна гарантировать отсутствие перевода стрелки занятой секции по вине системы, и это обеспечивается всем комплексом. При этом локальные устройства (например, безопасные блоки телеуправления ТУ-8Б и телесигнализации ТС-16Б) не используют понятия стрелок и секций, а выполняют только безопасную передачу сигналов (сохраняя при этом свой внутренний инвариант). Таким образом, на основании множества инвариантов подсистем формируется выполнение глобального инварианта, обеспечивающего безопасность работы АПК.

Применение самостабилизирующихся систем является универсальным и формализованным решением в отношении сбоев (*transient failure*; воздействий, которые изменяют состояние системы, но не её поведение), что актуально для СЖАТ [13].

Основными понятиями самостабилизирующихся систем являются сходимость (*convergence*) и замыкание (*closure*). Первое говорит о том, что система после старта должна гарантированно прийти к корректному состоянию за конечное время. Второе заключается в том, что если система находится в корректном состоянии, то она обеспечивает в дальнейшем переход только в другое корректное состояние. С точки зрения безопасных СЖАТ данные правила должны выполняться в случае отказов и сбоев.

Для верификации циклических систем, обладающих свойством самостабилизации, необходимо, чтобы во время этапа инициализации была достигнута сходимость, а во время циклического выполнения – замыкание.

Таким образом, если во время доказательства корректности происходит декомпозиция на этапы инициализации и циклического выполнения, то верификацию можно проводить на основании самостабилизирующихся свойств системы.

6 Условия, подлежащие выполнению для успешной верификации. Для того чтобы эффективно решать проблемы во время дедуктивного анализа циклических СЖАТ, обладающих свойством реактивности, необходимо, чтобы выполнялись следующие условия:

– наличие в исполняемом коде точки *State* (или множества точек), определяющей состояние системы;

– гарантированное прохождение выполнения программы через точку состояния системы за конечное время при любых внешних условиях и во всех рассматриваемых состояниях;

– выполнение всех функций цикла, подлежащих верификации, за конечное время.

Данные условия подлежат верификации посредством дедуктивного анализа. Другими словами, прежде чем перейти к верификации состояний системы и переходов между ними нужно доказать, что ПО действительно работает соответствующим образом.

На рисунке 6 показан пример циклического алгоритма, при котором в точке *State* можно определять состояние системы.

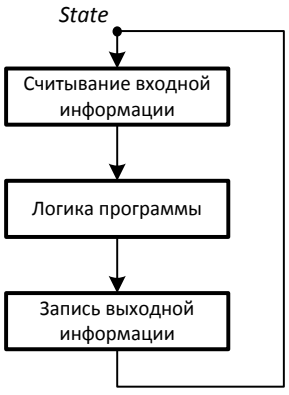

Рисунок 6 – Пример циклического алгоритма

Наличие точки *State* позволяет фиксировать момент завершения инициализации, а также формализовать и детерминировать моменты времени, когда состояние АПК изменяется качественно.

Если не гарантируется прохождение выполнения алгоритма программы через точку *State*, то переход из

одного состояния в другое может быть бесконечным. Если же во время длительного перехода из одного состояния в другое будет происходить запись выходной информации, то верификация такой модели будет крайне затруднительна из-за комбинаторного взрыва.

Если рассматривается с точки зрения верификации некоторое свойство АПК, и оно удовлетворяет только вышеописанным первым двум пунктам, то доказать его выполнение может быть затруднительно. Например, на рисунке 7 показан алгоритм, для которого не всегда может происходить проверка CRC памяти. Если на вход устройства будут поступать сообщения быстрее, чем оно может их обрабатывать, то точка *State* будет определена и выполняться на каждом витке цикла, однако одна из функций, ответственных за корректность состояния системы, никогда не выполнится.

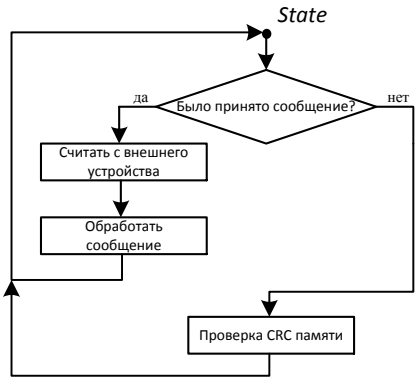

Рисунок 7 – Недостижимость верифицируемых функций

7 Переход к модели. Важным в последующей верификации циклической реактивной СЖАТ является создание на основе состояний и переходов между ними модели, в соответствии с которой работает АПК. Данный переход позволяет значительно упростить доказательство посредством проверки модели (*model checking*) с применением автоматической верификации, что возможно в случае, если система состоит из конечного числа состояний, и имеются в распоряжении вычислительные ресурсы. Кроме того, при верификации модели не является обязательным дедуктивный анализ.

Однако переход к модели подразумевает, что должны выполняться некоторые допущения [2]. Если же они не выполняются, то модель перестает отражать свойства системы. Рассмотрим допущения, которые характерны для циклических реактивных СЖАТ:

– представление внешних и внутренних значений передаваемых сигналов в терминах модели, т. е. преобразование из физического значения в абстракцию модели, не должно терять свойств, влияющих на доказательство корректности. Например, может быть потеряно свойство в случае преобразования внешнего непрерывного сигнала во внутреннее дискретное значение мгновенного состояния сигнала;

– представление времени в модели. Как таковое время в понятиях ПО отсутствует, и для решения этой проблемы значение времени привязывается к какому-либо физическому процессу, например в соответствии с тактами микропроцессора. Однако в этом случае необходимо учитывать, что в случае сбоев тактового генератора модель работает некорректно;

– предположение о нулевой задержке (*zero-delay hypothesis*). В данном случае считается, что считывание входной и запись выходной информации происходят мгновенно.

Так как алгоритм ПО АПК выполняется последовательно, то в реальности предположение о нулевой задержке полностью не выполняется. Более того, могут иметь место длительные операции, когда в течение некоторого времени считывается множество значений с АЦП и в дальнейшем вычисляется среднее как результат. Поэтому для минимизации проблем ПО строится таким образом, как показано на рисунке 6, и при этом считывание осуществляется как можно раньше, а запись выходных значений – как можно позже.

Для анализа времени переходов между состояниями удобно выделить понятие времени реакции системы, в течение которого возможен переход из одного состояния в другое. Определение пределов данного времени показано на рисунке 8.

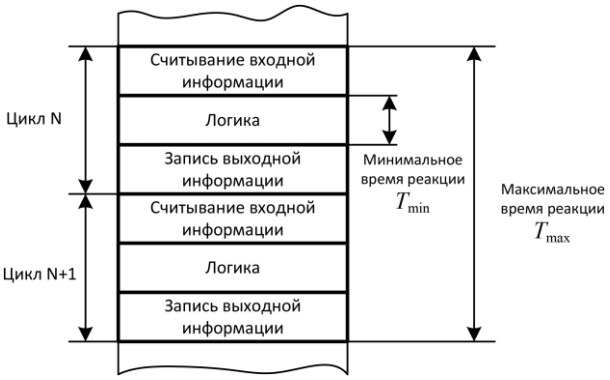

Рисунок 8 – Расчет времени перехода между состояниями

Данное время может быть быстро определено на основе анализа исходного кода программы, а практика показывает, что такого приближения для большинства СЖАТ достаточно [15]. Полученный параметр позволяет рассматривать модель системы, переходы между состояниями которой будут строго ограничены минимальным и максимальным предельным временем, что позволяет выполнить верификацию временных параметров функционирования.

Таким образом, переход к модели является отдельным этапом верификации, и его корректность должна быть доказана средствами дедуктивного анализа.

Заключение. Использование особенностей циклических и реактивных СЖАТ может значительно упростить разработку безопасных АПК, формализовать процесс доказательства корректности и повысить качество ПО посредством мероприятий на протяжении всего жизненного цикла. Понимание необходимых условий верификации и декомпозиция задачи доказательства корректности могут быть использованы как для разрабатываемых АПК на этапе проектирования, так и для верификации законченных устройств СЖАТ.

Список литературы

1 **Halbwachs, N.** Synchronous programming of reactive systems / N. Halbwachs // Springer. – 1993. – XIII. – 174 p.

2 **Charles, A.** Representation and analysis of reactive behaviors : A synchronous approach / A. Charles // Computational Engineering in Systems Applications IEEE-SMC, University of Nice Sophia Antipolis. – 1996. – P. 19–29.

3 **Locke, C. D.** Software architecture for hard real-time applications: cyclic executives vs. fixed priority executives / C. Douglass Locke // Journal Real-Time Systems. – Vol. 4, Is. 1. – 1992. – P. 37–53.

4 **Сивко, Б. В.** Доказательство корректности блока телеуправления 16-1 диспетчерской централизации «Нёман» / Б. В. Сивко // Вестник БелГУТа: Наука и транспорт. – 2012. – № 1 (24). – С. 18–21.

5 **Харлап, С. Н.** Верификация программного обеспечения микропроцессорной светооптической светодиодной системы / С. Н. Харлап, Б. В. Сивко // Вестник БелГУТа: Наука и транспорт. – 2012. – № 1 (24). – С. 22–25.

6 **Smith, David J.** Safety Critical Systems Handbook. A Straightforward Guide to Functional Safety, IEC 61508 and Related Standards, Including Process IEC 61511 and Machinery IEC 62061 and ISO 13849 / David J. Smith and Kenneth G. L. Simpson // Elsevier Ltd. – 2010. – 270 p.

7 **Сивко, Б. В.** Определение функции безопасности при верификации программного обеспечения микропроцессорных систем железнодорожной автоматики и телемеханики / Б. В. Сивко // Вестник БелГУТа: Наука и транспорт. – 2013. – № 1 (26). – С. 18–20.

8 **Hoare, C. A. R.** An axiomatic basis for computer programming / C. A. R. Hoare // Communications of the ACM. – 1969. – Is. 12(10). – P. 576–580, 583.

9 **Floyd, R. W.** Assigning Meaning to Programs / R. W. Floyd; ed. J. T. Schwartz // Mathematical Aspects of Computer Science. American Mathematical Society. – 1967. – P. 19–32.

10 **Бейбер, Р. Л.** Программное обеспечение без ошибок : пер. с англ. / Р. Л. Бейбер; под ред. Д. И. Правикова. – М. : Радио и связь, 1996. – 176 с.

11 **Кларк, Э. М.** Верификация моделей программ: Model checking : пер. с англ. / Э. М. Кларк; под ред. Р. Смелянского. – М. : МЦНМО, 2002. – 416 с.

12 **Вудкок, Д.** Первые шаги к решению проблемы верификации программ / Д. Вудкок // Открытые системы. – № 08. – 2006.

13 **Schneider, M.** Self-stabilization / Marco Schneider // ACM Computing Surveys (CSUR). – 1993. – Vol. 25, Is. 1. – P. 45–67.

14 **Dijkstra, E. W.** Self-stabilizing Systems in Spite of Distributed Control / Edsger Wybe Dijkstra // Communications of the ACM. – 1974. – Is. 11. – P. 643–644.

15 **Бочков, К. А.** Оценка временных параметров функционирования микропроцессорных устройств связи с объектами систем железнодорожной автоматики и телемеханики / К. А. Бочков, С. Н. Харлап, Б. В. Сивко // Вестник БелГУТа: Наука и транспорт. – 2012. – № 2 (25). – С. 12–15.

Получено 27.11.2013

S. N. Kharlap, B. V. Sivko. The verification of cyclic-executive railway systems.

The verification properties of cyclic-executive real-time railway systems with formal methods have been considered. It is shown that the proof of correctness separation into initialization and cyclic-executive phases allow to simplify the verification and provide an opportunity to use the model checking. The necessary conditions of proof with proof-theoretic approach has been formulated. Some ways of verification showed, which could check validation requirements and build a model based on the source code program.